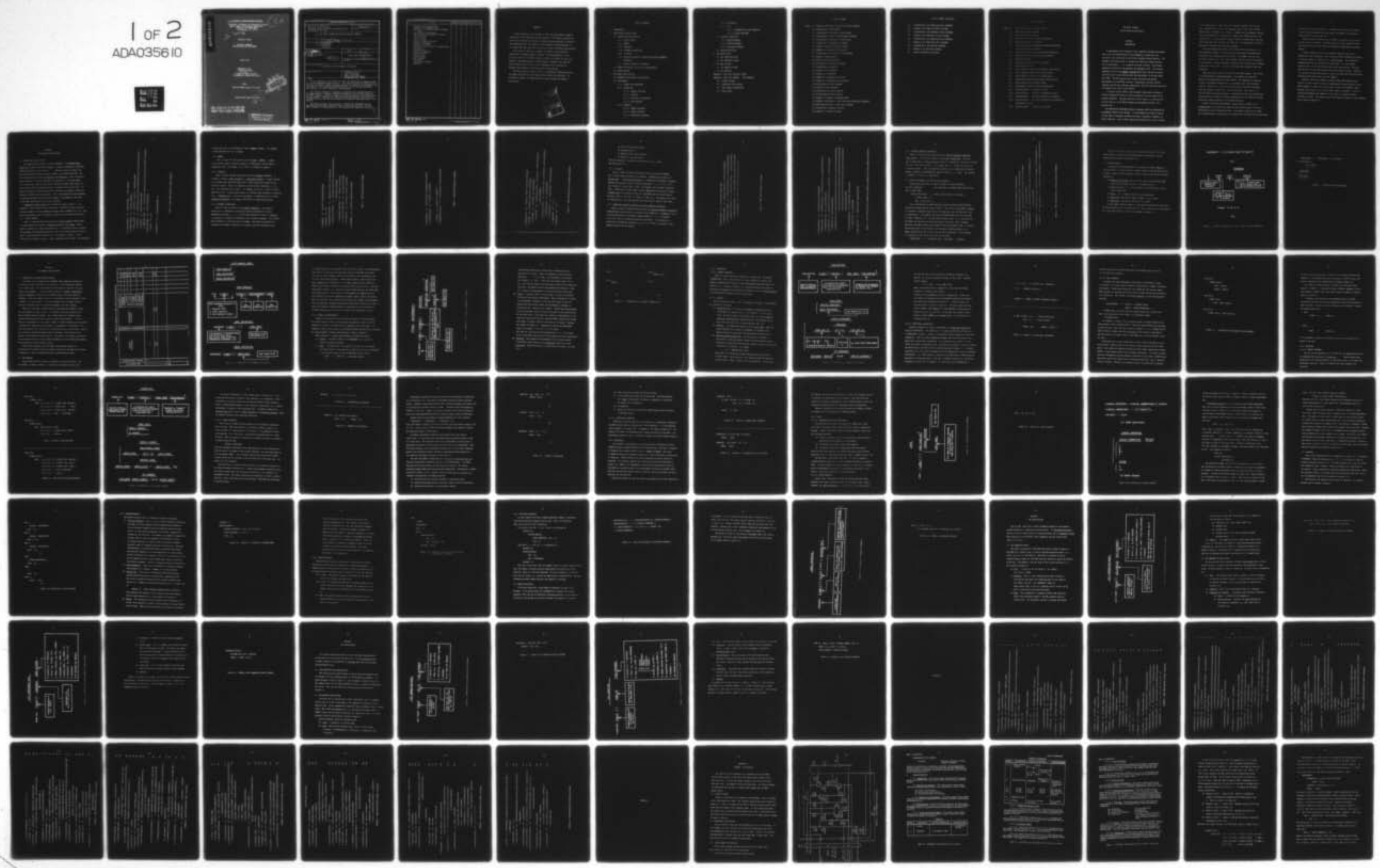
AD-A035 610 MOORE SCHOOL OF ELECTRICAL ENGINEERING PHILADELPHIA P--ETC F/G 9/2
THE NOPAL LANGUAGE SPECIFICATION AND USER MANUAL.(U)
AUG 76 H CHE, Y CHANG DAAA25-75-C-0650
UNCLASSIFIED 76-04 NL

1 OF 2
ADA035610



ADA035610

AUTOMATIC PROGRAM GENERATION PROJECT

Department of Computer and Information Sciences
Moore School of Electrical Engineering
UNIVERSITY OF PENNSYLVANIA
Philadelphia, Pa. 19174

6 10

New 410 039

TECHNICAL REPORT

THE NOPAL LANGUAGE SPECIFICATION AND USER MANUAL

August 1976

Submitted to the
Frankford Arsenal
U.S. Army
Bridge and Tacony Streets
Philadelphia, Pennsylvania 19137

Under

Contract Number DAAA-25-75-C-0650

Pa Univ. Phil Pa.

Moore School Report No. 76-04

14

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) University of Pennsylvania The Moore School of Electrical Engineering (D2) Philadelphia, Pa. 19174	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP

3. REPORT TITLE

6 The NOPAL Language Specification and User Manual.

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

9 Technical Report, - August 1976

5. AUTHOR(S) (First name, middle initial, last name)

10 Her-daw/Che
Yung/Chang

6. REPORT DATE 11 Aug[redacted]76	7a. TOTAL NO. OF PAGES 122	7b. NO. OF REFS
8a. CONTRACT OR GRANT NO. 16 DAAA25-75-C-0650 new b. PROJECT NO. 12 133p. c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) 14 Moore School Report 76-04	
	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	

10. DISTRIBUTION STATEMENT

Distribution of this document is unlimited

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Frankford Arsenal Dept. of the Army Bridge and Tacony Streets Philadelphia, Pa. 19137

13. ABSTRACT

As the maintenance cost continues to soar, the application of computer program for testing components becomes necessary. This report describes the specification and usage of automatic computer program generation for Automatic Test Equipment (ATE). The function of the automatic generated program is to validate UUT (Unit Under Test) design, test point availability, and packaging.

The system is referred to as NOPAL an acronym for Non-procedural Operational Performance Analysis Language. "Non-proceduralness" is the essential feature of the NOPAL language; in that test modules are specified modularly, and independently, one upon another. Therefore, the additions and/or modifications of tests can be incorporated easily and they are automatically attached in the proper places in the program.

The report consists of five sections. Introduction, EBNF Specification, Test Module Specification, UUT Specification, and ATE Specification. For each NOPAL statement, its EBNF definition, syntax diagram and examples are provided.

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6811

410 039

Security Classification

A-31408

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Automatic Test Systems (ATS)						
Automatic Test Equipment (ATE)						
Operational Performance Analysis Language (OPAL)						
Non Procedural OPAL (NOPAL)						
Abbreviated Test Language for Avionics Systems (ATLAS)						
Fault Location						
Fault Isolation						
Automatic Program Generation						
Automatic Programming						
Language Processing						
Electronic Circuit Analysis Program (ECAP)						
Determinacy						
Synchronization						
Sequencing Strategies						
Sequence Ordering						
Control Logic						
Directed Graphs						
Non-Procedural						
Incremental						
Structured Programming						
Simulation						

ABSTRACT

As the maintenance cost continues to soar, the application of computer program for testing components becomes necessary. This report describes the specification and usage of automatic computer program generation for Automatic Test Equipment (ATE). The function of the automatic generated program is to validate UUT (Unit Under Test) design, test point availability, and packaging.

The system is referred to as NOPAL, an acronym for Non-procedural Operational Performance Analysis Language. "Non-proceduralness" is the essential feature of the NOPAL language; in that test modules are specified modularly, and independently, one upon another. Therefore, the additions and/or modifications of tests can be incorporated easily and they are automatically attached in the proper places in the program.

The report consists of five sections. Introduction, EBNF Specification, Test Module Specification, UUT Specification, and ATE Specification. For each NOPAL statement, its EBNF definition, syntax diagram and examples are provided.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES (continued)

LIST OF TABLES

THE NOPAL LANGUAGE: SPECIFICATION AND USER MANUAL

SECTION 1
INTRODUCTION

As maintenance costs continue to soar, sometimes exceeding procurement costs, the cost effectiveness of using computers to reduce the time required to test mechanical and electrical equipment becomes apparent. The Automatic Test System (ATS), is designed to effectively reduce the major cost in maintenance --, i.e. diagnosis and fault location. The hardware component of the ATS is the Automatic Test Equipment (ATE). The software component of ATS is the computer programs which direct the ATE to execute specified tests by prescribing stimuli and measurement for the Unit Under Test (UUT). The ATE applies these stimuli and perform prescribed measurements at the UUT/ATE interface. Additionally, the ATE software component performs the necessary computations, makes the fault/safe decisions, and diagnoses the causes of the failure.

A software system, NOPAL or Non-Procedural Operational Performance Analysis Language Processor, has been developed for the automation of ATE program production. The main objective of this report is to describe the structure and use of the NOPAL language and programming Automatic Test Systems (ATS).

The Operational Performance Analysis Language (OPAL) was developed for programming Automatic Test Systems. It was designed to be used in testing a broad range of equipment including electronic, mechanical, hydraulic or optical machines. Also, an OPAL program can potentially be run on a variety

of ATE configurations. OPAL, high level computer language, like the high level language BASIC, begins with a keyword, such as CALL, CLOSE, DECLARE, DEFINE, GOTO, IF, REPEAT, etc. OPAL has a commonly used programming structure such as BEGIN_END block, IF_THEN_ELSE statement, CALL_RETURN function, GOTO transfer, etc. It also contains a number of built-in computational functions such as ABS, MAX, MIN, SIN, COS, LOG, SORT, etc. An OPAL Processor is being developed to interpret the OPAL source program.

The main disadvantage of OPAL arises from the fact that the user, i.e. the maintenance engineer or technician, must know computer programming and incorporate test sequencing into his program. For example, the user must explicitly specify, e.g., via a GOTO statement, the next test to be executed. He also must plan storage assignments. Moreover, OPAL requires many statements to describe a single test. In order to overcome these drawbacks, NOPAL was introduced.

There are several essential features of the NOPAL language. One of them is the methodology of non-proceduralness. Test modules are specified modularly and independently one after another. The sequencing of execution of tests is determined automatically in accordance with a fault locating strategy. Therefore, it is not necessary for users to provide information about the sequence in which test are to be executed. Also, there is no need for GOTO or CALL type statements. Sequencing execution and synchronizing each interactive statement are handled automatically by the automatic program generation software referred to as the NOPAL Processor.

Because of this non-proceduralness, another aspect of NOPAL is its' incrementality, in the sense that additions and/or modifications of the test specification can be incorporated easily. For example, when tests are added due to design changes, the user need not expend effort to modify the specifications,

but only needs to add the new part. The NOPAL Processor will then produce a new test program automatically and report the problems in the modified specification, if any, to the user.

Another important characteristic of NOPAL is that there is no need to specify storage assignments. The NOPAL Processor itself maintains several descriptive libraries which enables the sharing of common data by different modules.

A NOPAL statement can be either manually prepared by the user or generated by fault simulation. The collection of NOPAL statements describing a functional module is referred to as a specification. The information needed by NOPAL can be classified into three parts: (1) test modules specification, (2) UUT specification and (3) ATE specification. Section 2 describes the Extended Backus Normal Form Specification of NOPAL. Sections 3 to 5 provide a detailed description of each of the above three specifications, including formal definitions, uses and examples.

Appendix A provides a complete listing of NOPAL EBNF Specifications. Finally, in order to make it easier to understand the various aspects of the NOPAL language, a sample test specification, called "CPS #10559261", which performs fault analysis for a control power system is provided. It will be referenced throughout the remainder of this report to exemplify various statements in NOPAL. The complete source and formatted listings of "CPS #10559261" can be found in Appendix B.

SECTION 2
EBNF SPECIFICATION OF NOPAL

2.1 GENERAL EBNF SPECIFICATION

The formal syntax of NOPAL is given in Appendix A, in Extended Backus Normal Form (EBNF) specification language. Strings of characters, called non-terminals enclosed by angle-brackets < > represent syntactical names; and non-bracketed names represent terminal symbols, as in conventional BNF. The only two extentions to that are optionality and repetition. Units enclosed in square brackets [ ] indicate that they are optional, i.e., they may appear zero or one times. When an asterisk also follows the right square bracket]*, this indicates that the enclosed item may be repeated zero or more times. To improve readability of EBNF specification, the entries in Appendix A have been indented to reflect the various levels. Also, level numbers have been assigned to indicate the depth within the tree structure. Line numbers at the right of each EBNF specification are for easy reference.

Fundamental syntactic units used in NOPAL are shown on lines # 1 - 52, Appendix A. Most of them are self-explanatory. They have the usual syntactic structure common to other programming languages such as FORTRAN, PL/1, etc. Some brief descriptions as well as examples are given in the following sub-sections.

2.1.1 STRING CONSTANT

Table 2.1 shows the EBNF specification for string constant (STRING_CONST). A string constant can be either a character string or a bit string. Both of them are enclosed by a single quotation mark, '. A character string is composed of any number of the keyboard characters. But a bit string consists of only 0's and 1's, and ends with the character "B". For instance, 'ABC123', '-I5Q', '$*J1_56B' are character strings. '010'B, '10110'B are bit strings. All characters

```
1 <STRING_CONST>  ::=  <CHAR_STRING>  | <BIT_STRING>
  2 <CHAR_STRING>  ::=  '[<FULL_CHAR>]*'
    3 <FULL_CHAR>  ::=  <LETTER> | <DIGIT> | <SPECIAL_CHAR>
      4 <LETTER>  ::=  A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z|@|#|$|_
      4 <DIGIT>  ::=  0|1|2|3|4|5|6|7|8|9
      4 <SPECIAL_CHAR>  ::=  .|<|(|+|||&| |$|*|)|;|¬|-|/|,|%|_|>|?|:|#|@|'|=|"|BLANK
  2 <BIT_STRING>  ::=  '<BIT> [<BIT>]*'B
    3 <BIT>  ::=  0|1

1 <COMMENT>  ::=  /* [<FULL_CHAR>]* */
```

TABLE 2.1 EBNF SPECIFICATION OF STRING CONSTANT

enclosed by/* and */ are considered to form a comment statement. For example, /* HERE ENDS THE TEST */ is a comment.

2.1.2 NUMBER

Table 2.2 shows the EBNF specification for number (NUMBER). A number can be either signed or unsigned, integer or floating point and may appear in exponential form. For example, +1.8, -250.13, -21.38E+02 are numbers.

2.1.3 VARIABLES

Table 2.3 shows the EBNF specification for the variables (VARIABLE). A variable is either a single variable or a subscripted variable. A single variable is a variable name, which must begin with a letter and may be followed by more letters or digits. Twenty-six alphabetic and four special characters, @, #, $, and _, are considered to be letters. For example, E1, XYZ are correct variables while 3YQ is not. A subscripted variable is a variable name followed by a subscript list. A subscript list, enclosed by parentheses, is composed of one or several arithmetic expressions. For example, VAR (PI*2) is a subscripted variable.

2.1.4 ARITHMETIC EXPRESSIONS

Table 2.4 shows the EBNF for an arithmetic expression. An arithmetic expression is a combination of integers/variables and functions, with the mathematical operators +, -, *, /, and ** that evaluate to a value. A function is a function ID (identifier) followed by zero or several arguments. For instance, SINE (2*PI*K/N) is a function with one argument. The order of precedence for evaluating an arithmetic expression is in order of priority from high to low:

```
1 <NUMBER>  ::=  [<SIGN>] <UNSIGNED_NUMBER>
  2 <SIGN>  ::=  +|-
  2 <UNSIGNED_NUMBER>  ::=  <DECIMAL_NUMBER> [<EXPONENT>]
    3 <DECIMAL_NUMBER>  ::=  <UNSIGNED_INTEGER> [<DECIMAL_FRACTION>]
                          | <DECIMAL_FRACTION>
      4 <UNSIGNED_INTEGER>  ::=  <DIGIT> [<DIGIT>]*
      4 <DECIMAL_FRACTION>  ::=  .<UNSIGNED_INTEGER>
    3 <EXPONENT>  ::=  E[<SIGN>] <DIGIT> [<DIGIT>]

1 <INTEGER>  ::=  [<SIGN>] <UNSIGNED_INTEGER>
```

TABLE 2.2 EBNF SPECIFICATION OF NUMBER

```
1 <VARIABLE>  ::=  <IDENTIFIER> [(<SUBSCRIPT_LIST>)]
  2 <IDENTIFIER>  ::=  <LETTER> [<TAIL>]*

   3 <TAIL>  ::=  <LETTER> | <DIGIT>
  2 <SUBSCRIPT_LIST>  ::=  <ARITH_EXPR> [,<ARITH_EXPR>]*
```

TABLE 2.3 EBNF SPECIFICATION OF VARIABLE

```
1 <ARITH_EXPR>  ::=  [<SIGN>] <TERM> [<ADD_OP> <TERM>]*
  2 <TERM>  ::=  <FACTOR> [<MULT_OP> <FACTOR>]*
    3 <FACTOR>  ::=  <PRIMARY> [** <PRIMARY>]*
      4 <PRIMARY>  ::=  <UNSIGNED_NUMBER> | <VARIABLE> | <FUNCTION_CALL>
                        | (<ARITH_EXPR>)
        5 <FUNCTION_CALL>  ::=  <FUNCTION_ID> [(<ARGUMENT> [,<ARGUMENT>]*)]
          6 <FUNCTION_ID>  ::=  <IDENTIFIER>
          6 <ARGUMENT>  ::=  <ARITH_EXPR> | <STRING_CONST>
    3 <MULT_OP>  ::=  * | /
  2 <ADD_OP>  ::=  + | -
```

TABLE 2.4 EBNF SPECIFICATION OF ARITHMETIC EXPRESSION

(1) Unit enclosed in parentheses

(2) Exponentiation **

(3) Multiplication * and Division /

(4) Addition + and Subtraction -

Some good examples of arithmetic expressions are: x, (x + y)*Z, MAX (P**Q,Q/P), etc.

2.1.5 IF CLAUSE

Table 2.5 shows the EBNF specification for an if-clause (IF_CLAUSE). The if-clause causes execution of a statement depending on the value of a Boolean term. A relational expression is a combination of variables, constants, mathematical operators, and the relational operators (<, >, =, ¬, =, etc.) that evaluate to either TRUE or FALSE. For example, the relational expression (A*B) > (A**B) will be either TRUE or FALSE, depending on the values of A and B. A Boolean term is an expression that uses relational expressions and logical operators such as "¬" (NOT), "&" (AND), "/" (OR), etc. For example, (A>B | B>C) is a Boolean term. Note that Boolean terms also return a TRUE or FALSE value.

2.1.6 CONNECTION DIMENSION EXPRESSION AND VALUE DIMENSION EXPRESSION

Table 2.6 shows the EBNF specification for the connection dimension expression (CONN_DIM_EX), which consists of a connector and a dimension. A connector is one or several UUT point ID, enclosed by angle brackets "<", ">". For example, <J1-A, JX-Y> VOLT is a connection dimension expression. If the connector is replaced by an arithmetic expression, e.g., (Y + 5) *2 VOLT, it becomes a value dimension expression (VAL_DIM_EX).

```
1 <IF_CLAUSE>  ::=  IF <BOOLEAN_TERM> THEN
  2 <BOOLEAN_TERM>  ::=  <BOOLEAN_FACTOR> [V <BOOLEAN_FACTOR>]*
    3 <BOOLEAN_FACTOR>  ::=  <BOOLEAN_PRIMARY> [& <BOOLEAN_PRIMARY>]*
      4 <BOOLEAN_PRIMARY>  ::=  <RELATIONAL_EXPR> | ¬ (<BOOLEAN_TERM>)
        5 <RELATIONAL_EXPR>  ::=  <ARITH_EXPR> <RELATION> <ARITH_EXPR>
          6 <RELATION>  ::=  = | > | < | >= | <= | ¬= | ¬> | ¬<
```

TABLE 2.5 EBNF SPECIFICATION OF IF CLAUSE

```
1 <CONN_DIM_EX>  ::=  <CONNECTOR> [<DIMENSION>]
  2 <CONNECTOR>  ::=  <CONNECTOR_ID> | < <CONNECTOR_ID> [,<CONNECTOR_ID>]* >
    3 <CONNECTOR_ID>  ::=  <UUT_POINT_ID>
  2 <DIMENSION>  ::=  [<PREFIX>] <BASIC_UNIT> [/SEC/SEC] | <TIME_DIMENSION>
    3 <PREFIX> ::= <|G|M|C|U|N|P|MEG
    3 <BASIC_UNIT> ::= G|M|CD|CM|DB|FD|FT|GM|IN|HP|HZ|LB|L|I|LM|PC|AMP|BAR|DEG|ERG|GAL
                       |LUX|OHM|PPM|PSI|RAD|RPH|RPM|RPS|CU_M|DEGC|DEGF|DYNE|INHG|LINE
                       |MMHG|NEWT|SQ_M|STER|VOLT|WATT|CU_FT|FT_LB|HENRY|JOULE|IND_IP
                       |POUNDAL|BRAKE_HP|%
    3 <TIME_DIMENSION>  ::=  [<PREFIX>] <TIME_UNIT>
      4 <TIME_UNIT> ::= HR|MIN|SEC
1 <VAL_DIM_EX>  ::=  <ARITH_EXPR> [<DIMENSION>]
```

TABLE 2.6 EBNF SPECIFICATION OF CONNECTION DIMENSION EXPRESSION AND VALUE DIMENSION EXPRESSION

2.1.7 FUNCTION DIMENSION EXPRESSION

Table 2.7 shows the EBNF specification for function dimension expression (FUNC_DIM_EX). It is used in stimuli or measurement conjunctions. The basic unit of FUNC_DIM_EX is function primary, which is a function ID that may be followed by several functional arguments. For example, GEN (< = XYZ HZ, 'R1 FAILS') is a function primary. Function dimension expressions are a combination of function primary, constant, and mathematical operators such as +, -, * and /. For instance, 3 * SUPPLY (2 * PI HZ) is a FUNC_DIM_EX.

2.2 NOPAL TEST MODULES, UUT AND ATE SPECIFICATIONS

A high level definition of a NOPAL statement is shown beginning at line #53 in Appendix A. The following is an example taken from the top level of EBNF NOPAL:

```
< NOPAL_SPECIFICATION > ::=  [NOPAL] SPECIFICATION [< SPEC_NAME >];
        [< NOPAL_STMTS > ]*
        END  [< SPEC_NAME > ];
```

A test specification in NOPAL must begin with the keyword "SPECIFICATION", optionally prefixed with the keyword "NOPAL". Note that in string NOPAL language, if a keyword is composed of more than five characters, then only the first four are significant. For example, the keyword "SPECIFICATION" can also be written as "SPEC,' "SPECIFY," etc. The SPEC_NAME following "SPEC" is the identifier for the whole test specification and it may appear in the last "END" statement. Basically, the NOPAL structure corresponds to PL/1 "BEGIN_END" blocks. In between, there exists three classifications of statements as mentioned before, test module specification, UUT specification and ATE specification. This breakdown is illustrated by the second level EBNF specification.

```
<NOPAL_STMTS > ::=  < TEST_MODULE_SPEC >| < UUT_SPEC >  | < ATE_SPEC >
```

```
1 <FUNC_DIM_EX>  ::=  <FUNC_TERM> [<ADD_OP> <FUNC_TERM>]*
  2 <FUNC_TERM>  ::=  <FUNC_FACTOR> [<MULT_OP> <FUNC_FACTOR>]*
    3 <FUNC_FACTOR>  ::=  [<FUNC_MODIFIER> <MULT_OP>] <FUNC_PRIMARY>
      4 <FUNC_MODIFIER>  ::=  <UNSIGNED_NUMBER>
      4 <FUNC_PRIMARY>  ::=  <FUNCTION_ID> [(<FUNC_ARG> [,<FUNC_ARG>]* )]
                          | (<FUNC_DIM_EX>)
        5 <FUNC_ARG>  ::= [<RELATION>] <VAL_DIM_EX> | [=] <RANGE> | <STRING_CONST> | *
          6 <RANGE>  ::=  <VAL_DIM_EX>  +- <VAL_DIM_EX> | <VAL_DIM_EX> +- <ARITH_EXPR> [%]
```

TABLE 2.7 EBNF SPECIFICATION OF FUNCTION DIMENSION EXPRESSION

Note that the order of each specification and the order of the statements within each specification generally have no importance; the few exceptions are discussed in Section 3.1.1.

2.3 SYNTAX DIAGRAMS

In addition to NOPAL definitions, the usage of the NOPAL language is also described. In order to get the semantic flavor of the syntactical rules in EBNF, a graph of syntax diagrams of EBNF is given, utilizing the following conventions:

(1) Heading of the graph is placed in the center above the main graph and is the left hand side unit of a production rule.

(2) Syntactical unit (i.e., non-terminal) is indicated by all capital letters underlined.

(3) Tokens (i.e., terminal symbols) are not underlined.

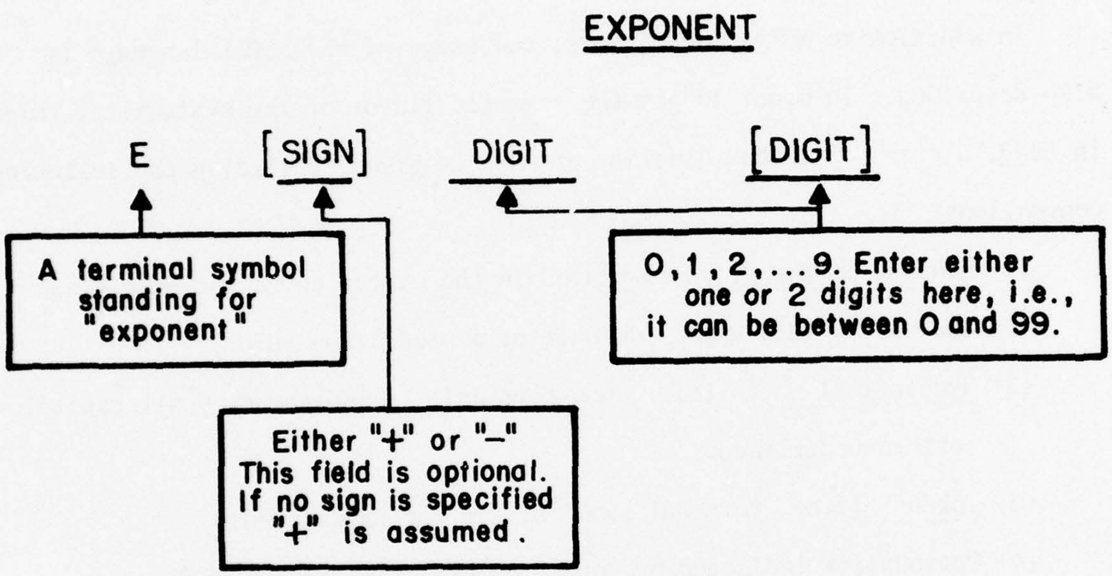
(4) Optionality indicated by square brackets [ ], as in EBNF.

(5) Repetition indicated by asterisk *, as in EBNF.

In the example shown in Figure 2.1, the boxes are optional and given some detailed explanations of the reference unit. Also, selectiveness is represented by a large left bracket as shown in the example in Figure 2.2.

<EXPONENT> ::= E [<SIGN>] <DIGIT> [<DIGIT>]

(a)



Example E+16, E–5

(b)

FIGURE 2.1. GENERAL ILLUSTRATION OF EBNF (a) AND ITS SYNTAX DIAGRAM (b)

< STRING_CONST > ::= < CHAR_STRING > | < BIT_STRING >

is represented as:

STRING_CONST

CHAR_STRING

BIT_STRING

FIGURE 2.2 ILLUSTRATION OF SELECTIVENESS

SECTION 3

TEST MODULES SPECIFICATION

3.1 OVERVIEW OF TEST MODULES SPECIFICATION

The inputs to the Automatic Test Equipment (ATE) production software are envisaged as coming from either of two sources: (1) the simulation software or (2) manual input. The test modules specifications are referred to as test steps or test modules. Each test module consists of stimuli, measurements, logic and diagnosis. These may be obtainable directly from the simulation component. These test modules can also be conveniently specified manually, using a tabular or a string format. For older equipment, test procedures already described in manuals need to be translated manually into this language.

Table 3.1 illustrates a tabular form for specifying test modules in which the step numbers are shown as well. Test modules are uniquely identified in the left hand column with labels. Also, unique labels are assigned to each stimuli and measurement portion of the test and to each diagnosis. A row in Table 3.1 corresponds to a test module stimuli (1) and measurement specification (2). A column on the right hand side of Table 3.1 corresponds to the diagnosis. The logic specification (3), placed at the intersection of the row and column, connects the respective stimuli/measurements with the diagnoses, which indicates the message (4) and operator interactions (5). The tabular form provides convenience in cases where there are several columns (diagnoses) per row (stimuli/measurements specification) or several rows per column.

Syntactically speaking, the test module specification consists of a collection of test module sections as shown in line #57 of the EBNF specification in Appendix A. They are further discussed in the following sections.

3.2 TEST MODULES

Usually there would be several test modules for testing each UUT. Each test module, as shown in Figure 3.1, consists of (in addition to the label)

TABLE 3.1 TABULAR APPROACH TO TEST MODULE SPECIFICATIONS

TEST MODULE ID	STIMULI CONJUNCTION ID	STIMULI FUNCTIONS	MEASUREMENT CONJUNCTION ID	MEASUREMENT FUNCTIONS						
1	1		1							
		1		2					3	
m			n							
m+1	2		1							
n+m			n							

DIAGNOSIS ID							
OPERATOR RESPONSE						5	
OPERATOR MESSAGE	TIMING					4.4	
	TYPE					4.3	
	OTHER PARA-METERS					4.2	
	COMPON-ENTS AFFECTED					4.1	

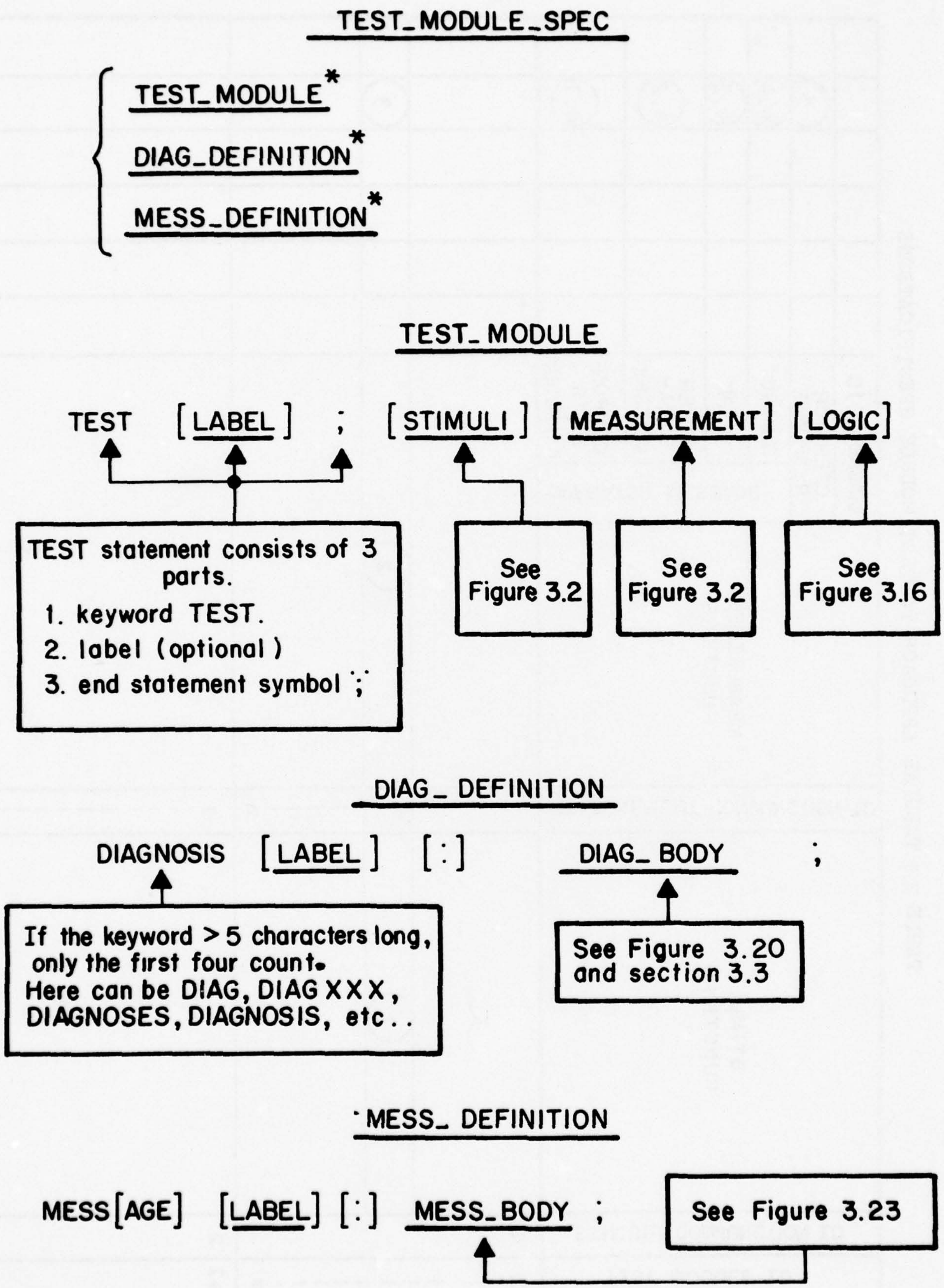


FIGURE 3.1. ILLUSTRATION OF TEST MODULE SPECIFICATION

(1) stimuli that need to be applied to the UUT at test time, (2) the measurements that need to be taken with the comparisons that will determine the results, (3) logic to select diagnoses based on the results of the measurements, and (4) the corresponding diagnoses. Semantically speaking, stimuli specify the pins, connections and functions, e.g., DC power supply function, needed to generate the stimuli waveforms to be applied to the respective pins. Then some measurement functions, e.g., testing constant voltage function, are taken under some specified pins. The results of the measurement are communicated back and evaluated to select the appropriate diagnoses, which indicate the failure modes in a message. The logical operators connect the stimuli and measurements with the appropriate diagnoses and operator interaction as specified in Table 3.1. These four components mentioned above are considered to be the central core of NOPAL language and are discussed in detail in the subsequent sections.

3.2.1 STIMULI AND MEASUREMENTS

Stimuli and measurements specifications are similar in syntax although they differ greatly in their interpretation and semantics. Both of them are composed of two parts (1) labels which are optional and (2) waveforms. Consequently, common syntax is used for specifying stimulus and measurement waveforms. The syntax diagram of stimuli (measurements) is shown in Figure 3.2.

Major components of stimuli (measurements) are exemplified below:

(1) Keyword. A keyword "STIMULI" (or "MEASUREMENT") is required to specify the type of statement.

(2) Label. An unique label is assigned to a stimuli (or measurement) for identification. Syntactically, a label can be either an identifier or an unsigned integer as shown in EBNF specification below:

< LABEL > ::= < IDENTIFIER > | < UNSIGNED_INTEGER >

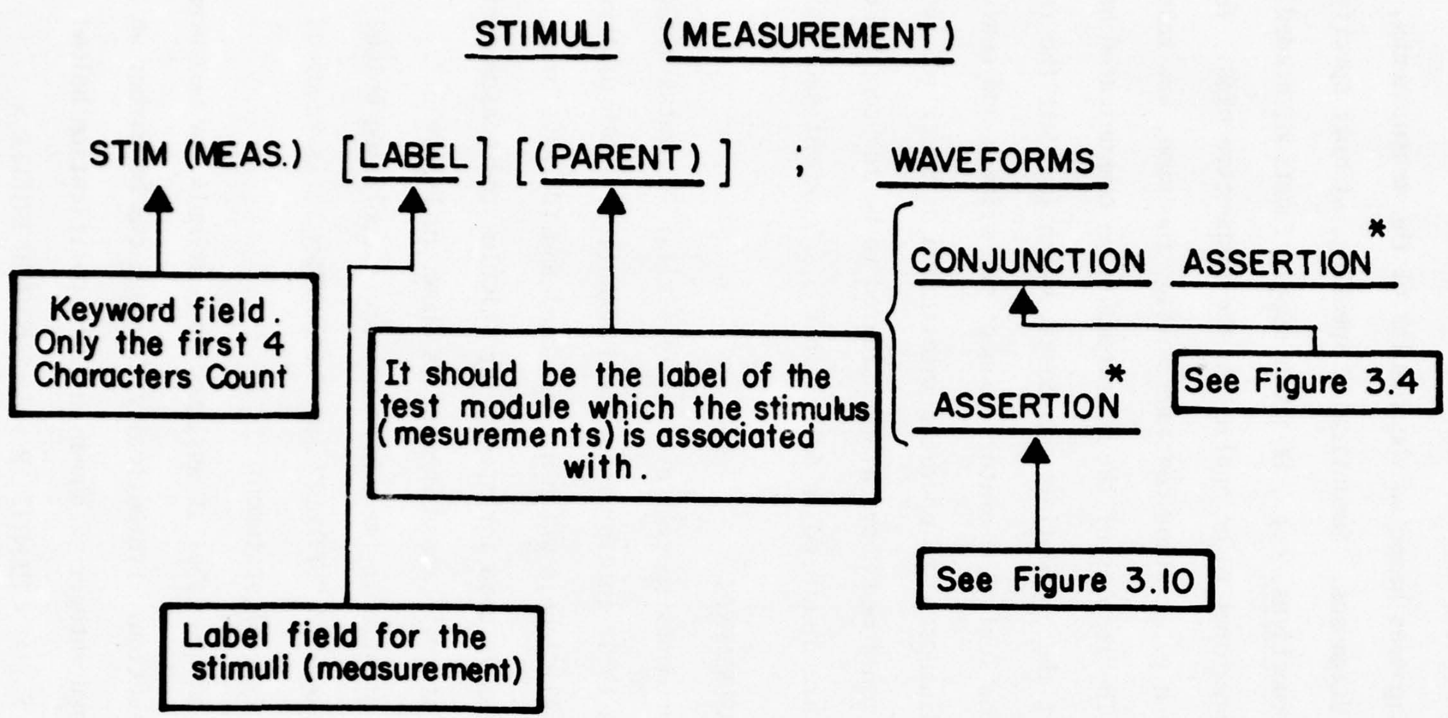


FIGURE 3.2 ILLUSTRATION OF STIMULI (MEASUREMENTS) STATEMENT

An identifier begins with a letter and is followed by zero or more letters or digits. Twenty-six alphabetic and four special characters, @, #, $, and _, are considered to be letters. For instance, '3', 'B5_J', '#5@T' are valid labels, while '3X' is not. Note that the label is not necessarily required to be explicitly specified. The NOPAL Processor will automatically generate a label for a stimuli (or measurements) if the user does not provide one.

(3) Parent. A "parent", which is also a label, is used as a qualifier to indicate the parental relationship. Thus, the parent of a stimuli (or measurements) statement is just the label (name) of the test module which the current stimuli (or measurements) statement is associated with. Therefore, the relation ensures the non-proceduralness and statements then can be put at any place in the specification. On the other hand, if the parent of a stimuli (or measurements) is not explicitly specified by the user, then the NOPAL Processor will assume and assign by default, the last test label as its parent. Consider the example in Figure 3.3. Situations (a) and (b) are equivalent; namely, the stimuli 'Z' has test 'Y' as its parent.
Note that the restriction on non-proceduralness, i.e., if no parent field is specified, the position of the statement does have significance.

(4) Waveforms. Both stimuli and measurements have the same structure of waveforms, which consist of one conjunction and/or one or more assertions. Detailed discussion is given below in Sections 3.2.2 and 3.2.3.

```
TEST X;
     .
     .
     .
TEST Y:
     STIMULI Z:
          .
          .
          .
```

(a)

```
TEST X;
     STIMULI Z(Y);
          .
          .
TEST Y;
     .
     .
     .
```

(b)

FIGURE 3.3. ILLUSTRATION OF THE USAGE OF PARENT FIELD

3.2.2 CONJUNCTION

3.2.2.1 GENERAL STRUCTURE

Figure 3.4 depicts the syntax diagram of a conjunction. The keyword (CONJUNCTION), label, and parent have the same syntactical structure and semantic interpretation as that of stimuli/measurements described earlier in Section 3.2.1. As shown in Figure 3.4, a conjunction body can be either a collection of triplets (to be described in Section 3.2.2.2) or a back-reference (to be described in Section 3.2.2.4), but not both.

3.2.2.2 TRIPLETS

The main conjunction body is just a collection of triplets. The components of a triplet are basically:

(1) Connection Point. It specifies the names of connecting points of an UUT which a triplet waveform is to be applied to or measured from. The specification of connection points is called connection dimension expression where dimension has the usual meaning such as Hertz (Hz), Kilovolts (KV), or micro-seconds (μSEC), etc.

(2) Relation. In conjunction triplets, only an equal sign = should be used to indicate the relation of assignment, as opposed to the Boolean relation in assertions to be discussed in Section 3.2.3.2.

(3) Function Dimension Expression. The third part of a triplet is specified by a function dimension expression which does not include references to connection points, but must include at least one term with a reference to a function. For example,

< A, B > = ACM(VAR2 RMS, 50 Hz)

where ACM is an alternating current measuring function with 50 Hz as input value. After executing, this function assigns the measured value to VAR2, a target variable to be used elsewhere. In this case,

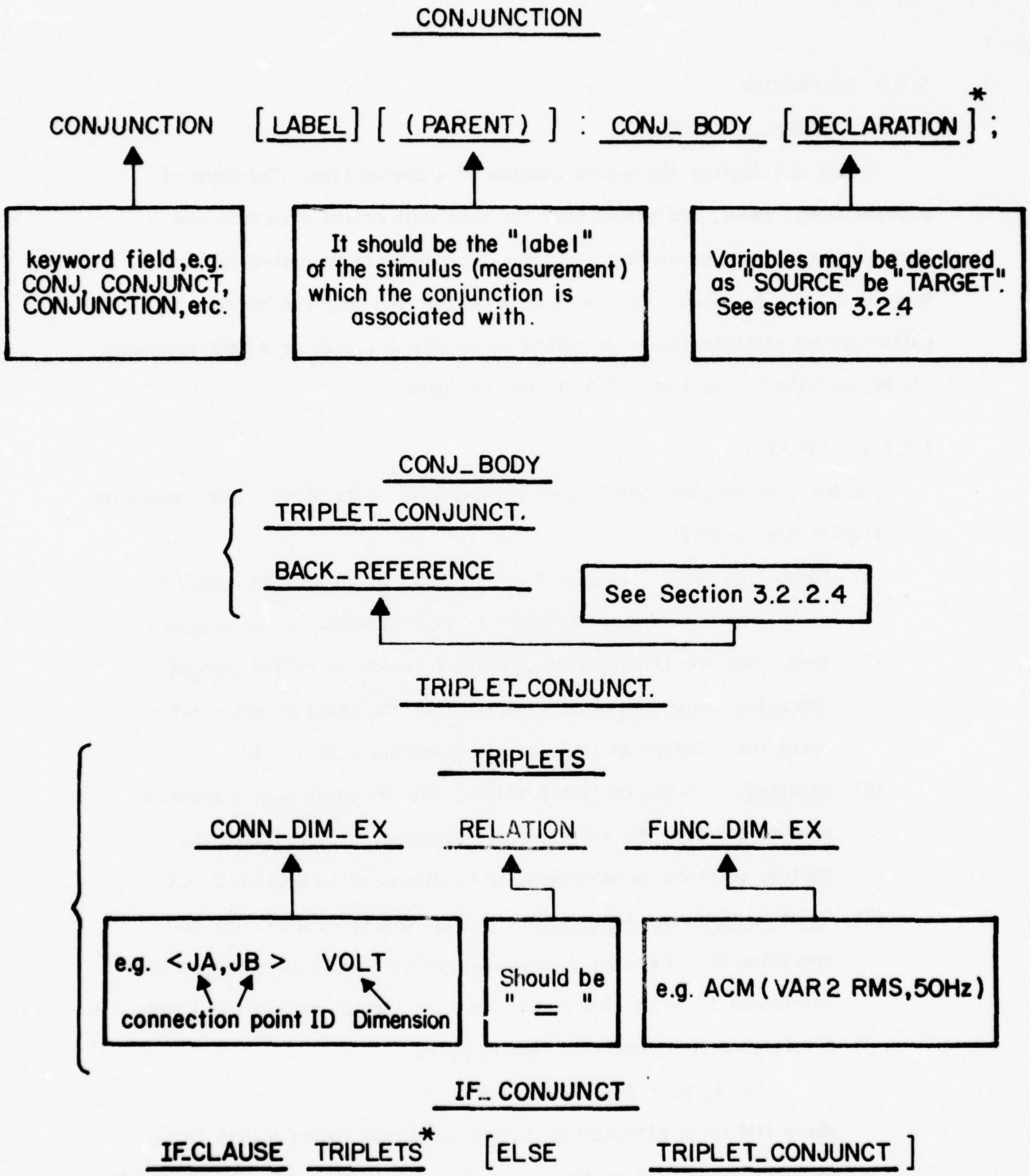


FIGURE 3.4 ILLUSTRATION OF CONJUNCTION STATEMENT

the function ACM is not considered as an implicit condition. By default, this kind of conjunction returns a "true" value. Consider another example:

< A, B >= ACM (< 7 volts RMS, 50 Hz)

where the measured value is compared to 7 volts and a true/false condition is returned to the triplet.

Triplets are the fundamental units of a conjunction. If more than one triplet are required, they are optionally enclosed in parentheses and connected by conjunction operator "&". A typical stimuli conjunction triplet is shown in Figure 3.5. A DC power supply function called P_SUPPLY, with arguments 29 V and 1000 MA is to be applied to connection points J1_G and J1_B. At the same time, another stimuli function, called COOLANT, with argument 14° F. is to be applied to a connection point S1.

3.2.2.3 CONDITIONAL CONJUNCTION

Another alternative for triplet conjunction is a conditional conjunction, called IF_CONJUNCTION. (See Figure 3.4). It looks like the PL/1 or ALGOL type "IF_THEN_ELSE" structure. Figure 3.6 illustrates a typical example of conditional conjunction. VAR is an input independent variable that has been evaluated elsewhere and is used by this conjunction. It is referred to as a SOURCE variable. IE29VL and T which are measured in this conjunction are declared as TARGET variables. The declaration of variables will be further discussed in Section 3.2.4. Boolean expression, VAR < 20, is first evaluated to set the condition for selecting different measurements, i.e. AMP_M function, applying to J1_G UUT point or TEMP_M function, applying to S1 connecting point. Note that, as shown in Figure 3.4 in the syntax diagram, or in lines #76-77 of Appendix A in EBNF form, the IF_CONJUNCTION is

```
( < J1_G, J1_B >    V = P_SUPPLY (29 V, 1000 MA)) &

( < S1 >    = COOLANT (14 DEG F));
```

FIGURE 3.5. EXAMPLE OF STIMULI CONJUNCTION TRIPLET

```
IF VAR < 20 THEN ( < J1_G >    = AMP_M (IE29VL MA))
            ELSE ( < S1 >    = TEMP_M (TDEGF))

            SOURCE:  VAR          TARGET:  IE29VL, T;
```

FIGURE 3.6 EXAMPLE OF A CONDITIONAL CONJUNCTION

defined recursively and nested conditions are thoroughly legal as in PL/1 or LISP high level languages.

3.2.2.4 BACK REFERENCE

Frequently the same measurement (conjunction) is specified in several test modules or a single stimuli waveform is specified for a number of different test modules. Without duplicating the same conjunction over and over again, NOPAL provides a convenient way for the users to handle the multiple occurrences of a conjunction. This feature, called back reference, has the following EBNF structure.

```
< BACK_REFERENCE > ::=  [SAME] AS < STIM_MEAS_LABEL >
                        [EXCEPT  < SIMPLE_CONJUNCTION > ]
```

If EXCEPT part is missing, then the current conjunction is exactly the same as the conjunction in the stimuli (or measurement) named < STIM_MEAS_LABEL >.

For example, in Figure 3.7 the conjunction body of stimuli S55112 has the same context as that of stimuli S55111. Note that "Back Reference" has no effect on the assertion. Thus, the last statement of Figure 3.7, i.e., "STIMULI S55113: SAME AS S55111;" implies that the conjunction of that stimuli is just BODY_A. But the assertion of stimuli S55113 is not BODY_B and is not yet defined or does not exist.

Occasionally two or more conjunctions are just slightly different in one or more triplets. Hence, we may use the feature of back reference with the EXCEPT option. Each connection dimension expression of the simple conjunction following EXCEPT is tested against that of the referenced conjunction. If a match is found, then the corresponding function dimension expression of the triplet in EXCEPT body substitutes that of the triplet in the referenced conjunction. Thus, a modified triplet is formed. Otherwise, the unmatched triplet in EXCEPT body is appended

```
TEST 55111;
     STIMULI S55111;
          CONJ:  (BODY_A)
          ASSERT:  (BODY_B)

TEST 55112;
     STIMULI S55112;
          CONJ:  SAME AS S55111;

TEST 55113;
     STIMULI S55113:  SAME AS S55111;
```

FIGURE 3.7. ILLUSTRATION OF THE USAGE OF BACK REFERENCE

to the referenced conjunction. Figure 3.8 is an example excerpted from Appendix B. The language processor will catch the "EXCEPT" feature and create a conjunction for stimuli S55131 as shown in Figure 3.9.

Note that at connection points, J1_H and J1_B, the same stimuli function, P_SUPPLY, is applied but with a different argument, 10 MA is used instead of 100 MA. And a new stimuli function, LOAD_NL is added to points J1_E and J1_B.

In summary, back reference in one statement refers to another conjunction waveform which has been predefined or which will be defined in other statement. However, cyclical references are not allowed. Consider the example shown below.

```
TEST;
        STIMULI   X:       SAME AS Y;
                  .
                  .
                  .
TEST;
        STIMULI   Y:       SAME AS X;
```

It is considered illegal by the processor and an error message will be output to the user.

3.2.3 ASSERTION

3.2.3.1 GENERAL STRUCTURE

The role of the assertions is tc perfrom the pure computation necessary to supplement the facilities of conjunctions. Unlike conjunctions, assertions do not include references to connection points or to stimuli and measurement functions. Figure 3.10 depicts the syntax diagram of an assertion.

```
TEST 55111;
     STIMULI S55111;
          CONJ: < J1_G, J1_B > V = P_SUPPLY (29V, 1000 MA) &
                < J1_F, J1_B >V = P_SUPPLY (15V,      1 MA) &
                < J1_H, J1_B >V = P_SUPPLY (24V,    100 MA) &
                < J1_A, J1_B > =  LOAD_L   ( 2200 OHM);

TEST 55131;
     STIMULI S55131:
          CONJ:  SAME AS S55111  EXCEPT
                < J1_H, J1_B > V = P_SUPPLY (24V, 10 MA) &
                < J1_E, J1_B >   = LOAD_NL;
```

FIGURE 3.8 EXAMPLE OF BACK REFERENCE

```
TEST 55131;
     STIMULI S55131;
          CONJ:  < J1_G, J1_B > V = P_SUPPLY (29V, 1000 MA) &
                 < J1_F, J1_B > V = P_SUPPLY (15V, 1 MA) &
                 < J1_H, J1_B  >V = P_SUPPLY (24V, 10 MA) &
                 < J1_A, J1_B >   = LOAD_L (2200 OHM) &
                 <  J1_E, J1_B  > = LOAD_NL
```

FIGURE 3.9. THE EFFECT ON THE BACK REFERENCE

ASSERTION

ASSERTION [LABEL] [(PARENT)] : ASSE_ BODY [DECLARATION]* ;

keyword field, e.g.
ASSE, ASRT, ASSERT
ASSERTION, etc.

It should be the "label"
of the stimulus (measurement)
which the assertion is
associated with.

"SOURCE" or "TARGET"
variable declaration
See Section 3.2.4

ASSE_ BODY

{ SIMPLE_ASSERT.
{ IF_ASSERT.

SIMPLE_ASSERT

RELATIONAL_EXPR

ARITH_EXPR. RELATION ARITH_EXPR.

WITHIN_ EXPR.

ARITH_ EXPR. = ARITH_ EXPR. + − ARITH_ EXPR. [%]

IF_ ASSERT

IF_CLAUSE SIMPLE_ASSERT. [ELSE ASSERT_ BODY]

FIGURE 3.10 ILLUSTRATION OF ASSERTION STATEMENT

The keyword "ASSERTION" (or ASSE, ASSERT) must be stated first. Label, parent and declaration all have the same structure as in conjunction. Unlike conjunction, assertion body does not include the feature of back reference. Hence, if BACK_REFERENCE appears at the assertion's parent level, i.e., stimuli/ measurement, it applies to the conjunction only. As shown in Figure 3.10, assertion body can be either a simple assertion or a conditional assertion. They are further discussed in the following subsections.

3.2.3.2 SIMPLE ASSERTION

Basic form of a simple assertion consists of two arithmetic expressions and a relation. EBNF specification of a relation is shown in Figure 3.11. It can be equal "=", greater than ">", less than "<" or combination of them (with negation "¬"). There exist two different interpretations of a simple assertion. When an assertion is used purely to indicate the value of variables, it should follow the form:

TAR_VAR = ARITH_EXPR

where the arithmetic expression on the right hand side is first evaluated, and then the result is assigned to the variable "TAR_VAR". Here the equal sign acts as an assignment symbol. It is essential that TAR_VAR be declared as TARGET variable and be the only variable on the left hand side of the equal sign. See Figure 3.12 for an example.

Note that LOG is a target variable and LOG10 is an evaluation function (to be further discussed in Section 5.2). Stimuli and measurement assertions, while utilizing the same syntax, are interpreted differently by the automatic test programming system. The stimuli assertions are interpreted in order to generate some data. Hence, they have only the form TAR_VAR = ARITH_EXPR and the meaning as described above.

RELATION ::= =|>|<|>=|<=|¬=|¬>|¬<

FIGURE 3.11. ILLUSTRATION OF "RELATION"

ASSERT A1: LOG = 20*LOGIC (V1/3.981E-6)

TARGET: LOG SOURCE: V1;

FIGURE 3.12. EXAMPLE OF AN ASSERTION

Measurement assertions can also play the role of descriptions of conditions, as if prefaced by "IF". This kind of measurement assertions return true/false values. Arithmetic expressions on both sides of the relationship (see Figure 3.10) are evaluated and a Boolean decision is made. Variables cannot be declared as "TARGET" in this case. Figure 3.13a gives an example of this kind of assertion. Note that stimuli/measurements/failures functions cannot be used in assertions.

Another syntax form to express a Boolean assertion is of the structure:

< ARITH_EXPR > = < ARITH_EXPR >+- < ARITH_EXPR > [%]

Thus, the example in Figure 3.13a can be written in the form shown in Figure 3.13b or 3.13c. In each case, if IE29V (calculated elsewhere) lies between 37 and 57, the assertion returns a "true" value.

By default, measurement assertions that generate data also return a "true" value. In this way, every measurement assertion possesses either a true or a false value. The conjunctive value of the measurement assertions and of measurement conjunction is considered to be the value of this measurement. Then the logical operator (to be described in Section 3.2.5.2) will select diagnoses based on the true/false value(s) returned by executing the measurements of a test module as described in Section 3.1 and Table 3.1.

Note that the number of assertions in a stimulus or measurement may vary. They may be arbitrarily ordered by the user (non-proceduralness). The NOPAL Processor will decide, based on the declaration of variables, the correct executing sequence among those assertions and conjunctions. Declaration is further discussed in Section 3.2.4. Finally, the main restrictions on assertions (as compared to conjunctions) are summarized below:

(1) Assertions must not include references to connection points.

(2) Stimuli/measurements/failures functions cannot be used for assertions.

(3) Assertions do not have a "back-reference" feature.

ASSERTION: ABS (IE29V - 47) <= 10
SOURCE: IE29V;

(a)

ASSERTION: IE29V = 47 +- 10
SOURCE: IE29V

(b)

ASSERTION: IE29V = 47 +- 21.3%
SOURCE: IE29V

(c)

FIGURE 3.13. EXAMPLES OF ASSERTIONS

(4) Each assertion has at most one target variable.

(5) A test module may contain several assertions. But each assertion is a single "triplet-like" structure, as opposed to a collection of triplets in a conjunction.

(6) No dimensions

(7) Stimuli assertions are exclusively computational and do not return a true/false value.

3.2.3.3 CONDITIONAL ASSERTION

Conditional assertions have the same structure as a conditional conjunction. The EBNF specification is listed on line #84 of Appendix A. Its corresponding syntax diagram is in Figure 3.10. An example is also shown in Figure 3.14. The condition "IF VAR1 = 60" determines which simple assertion is to be taken. The appropriate true/false value is returned according to the chosen simple assertion.

3.2.4 DECLARATION

All variables in conjunctions or assertions should be declared to improve readability and to promote modularity of the specification. The user is required to designate the variables into two classes, SOURCE and TARGET. The values of SOURCE variables are determined elsewhere, in other assertions or conjunctions or diagnoses. TARGET variables are locally evaluated and may be referred to elsewhere. For instance, consider the conjunction and assertion waveforms in Figure 3.15. AMP_M is an "amperemeter" function which measures the current in milliampers at connection point J1_B and determines the value of variable IE29V. Then the assertion that follows refers to IE29V as source and can make true/false decisions for the assertion.

Declaring variables not only facilitates the readability, but more importantly,

```
ASSERTION   $W_03:
      IF VAR1 = 60 THEN   F1 = 5 * 1E+06 + 60
                   ELSE   F1 = 5 * 1E+06 + 2.5

      SOURCE:   F1, VAR1;
```

FIGURE 3.14. EXAMPLE OF A CONDITIONAL ASSERTION

```
CONJUNCTION:  J1_G MA = AMP_M (IE29V MA)
      TARGET:   IE29V;

ASSERTION:   ABS (IE29V - 47)  <= 10
     SOURCE:   IE29V;
```

FIGURE 3.15. EXAMPLE OF A CONJUNCTION AND AN ASSERTION

the inherent information of declaration is used by the language processor to determine the sequencing of test execution. Note that assertions specified in stimuli or measurements can be executed before or after the respective conjunctions but not concurrently with the conjunctions.

Whenever variables are not explicitly declared as SOURCE or TARGET, by default they are considered to be SOURCE variables.

3.2.5 LOGIC

3.2.5.1 INTRODUCTION TO LOGIC FUNCTIONING

The main objective of the ATE system is to examine Unit Under Test (UUT), to diagnose faulty components and to output an appropriate message. "Logic", the decision stage of the test modules, carries out the following functions:

(1) Selects a diagnosis based on the true/false value returned by executing the measurements of a test module.

(2) Facilitates test modules sharing the diagnosis

(3) Facilitates interactive communication with the operator

Details of the logic operators are further discussed in the following Subsection 3.2.5.2. The syntax diagram for Logic is shown In Figure 3.16.

The main body, called LOGIC_DIAG_LIST, is a list of diagnosis information which consists of a logical operator followed by a diagnosis label. A typical example is shown in Figure 3.17, where d_i is a diagnosis lable (corresponding to the label to be discussed in Section 3.3) and */V/&/¬ are logical operators to be discussed below.

3.2.5.2 LOGICAL OPERATOR

Figure 3.18a illustrates the EBNF specification and the syntax diagram of the logical operators can be seen in Figure 3.18b. Broadly speaking, the <u>logical connective</u> (|, &, |¬, &¬, *) indicates when a

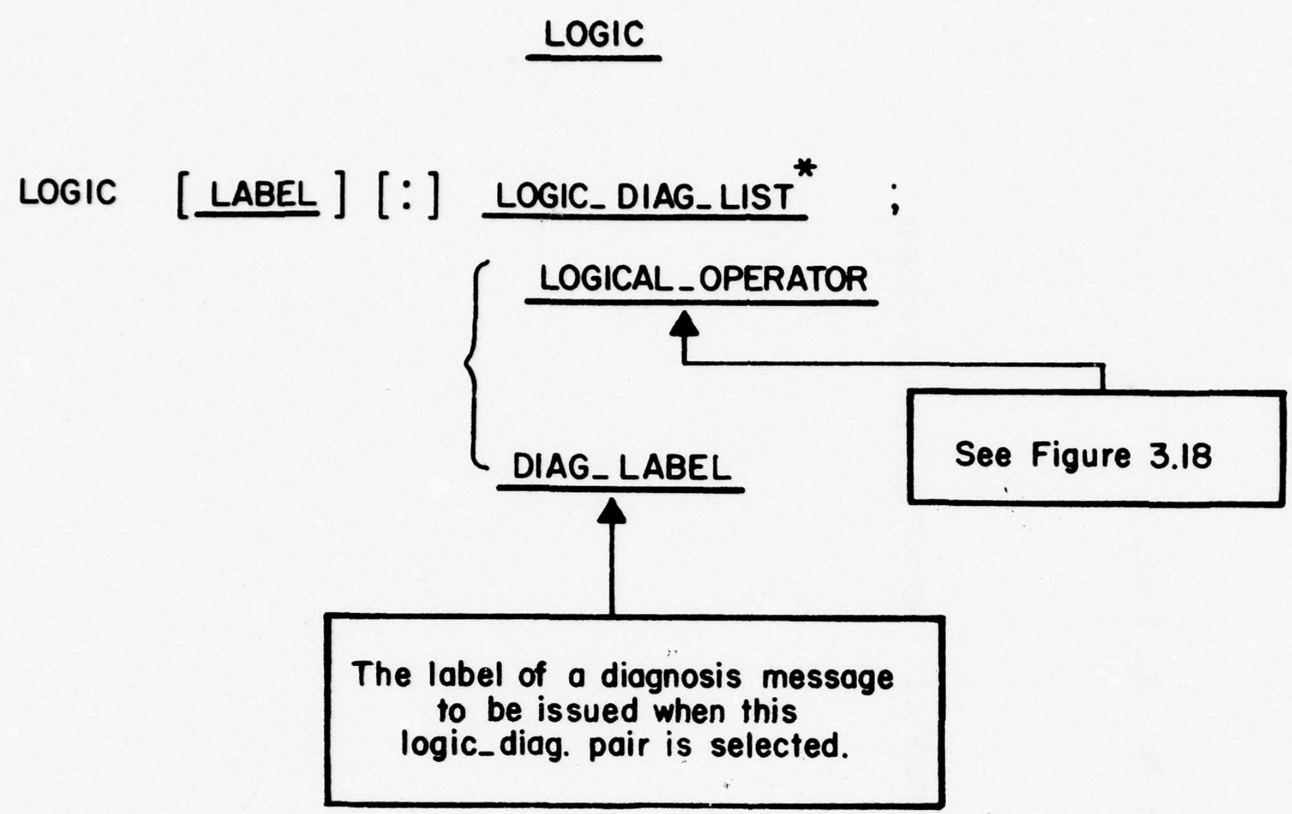


FIGURE 3.16 ILLUSTRATION OF LOGIC STATEMENT

LOGIC: $*d_1$, $|d_2$, & $\neg d_3$;

FIGURE 3.17. EXAMPLE OF A LØGIC STATEMENT

<LOGICAL OPERATOR> :: <LOGICAL_CONNECTIVES> | <AFTER>

<LOGICAL_CONNECTOR> :: = I | I¬ | & | &¬ | *

<AFTER> :: = A | A¬

(a) EBNF Specification.

LOGICAL OPERATOR

LOGICAL CONNECTIVE [AFTER]

I
I¬
&
&¬
*

AFTER

A
A¬

(b) Syntax Diagram.

FIGURE 3.18 ILLUSTRATION OF LOGICAL OPERATOR

stimuli-measurement pair selects a diagnosis. "After" connectives are used for reverse order selection, where a diagnosis selects a stimuli-measurements pair.

A disjunction operator (|) is to connect any one stimuli-measurement pair with a selected diagnosis, where the measurement returns a "true" value. A negation symbol is added ($|\neg$) to indicate the specified diagnosis is selected when the measurement returns a "false" value. For instance, let $B(M_i)$ be the Boolean value returned by test i and the corresponding logic look like:

LOGIC: $|d_1$, $|\neg d_2$, $*d_3$;

First, $B(M_i)$ is evaluated. Then if it turns out to be true, diagnosis d_1 is selected, otherwise d_2 is issued. However, d_3 is selected regardless of the returned value. In other words, a "*" symbol is used when the diagnosis selection is independent of the returned value. Note that the use of | and $|\neg$ are similar to those of the words "THEN" and ELSE", respectively, in an "IF" type statement in programming language. The above example can be rewritten by PL/1 - like language as follows:

```
SELECT d3;
IF B(Mi) THEN SELECT d1;
        ELSE SELECT d2;
```

The conjunction symbols (&, &$\neg$) are used similarly to indicate that the conjunction of true/false values, returned from execution of measurements in a number of stimuli-measurements pairs, are needed in order to select a diagnosis. Consider the following example in Figure 3.19. Suppose diagnosis d_i is referred to only in tests 1, 2 and 3. Then d_i will be selected if and only if the values of measurements 1 and 3 are true and measurement 2 returns

false. Its PL/1 -like structure appears as follows:

$$\text{IF } B(M_1) \ \& \ \neg B(M_2) \ \& \ B(M_3) \text{ THEN SELECT } d_i;$$

Note that a */|/|¬ -type logical operator can be handled by one pair of stimuli-measurements at a time, while &/&¬ -type operators have to do with all the related pairs.

Another type of logical operators, "AFTER" and "AFTER-NOT", shows the reverse order selection, where a diagnosis selects a stimuli-measurement pair. The symbols ? (after) and ?¬ (after-not) are used for this purpose to indicate that the stimuli-measurements pair is to be executed immediately following the execution of diagnosis part. This feature is intended to accomodate interactive communications between the computer and the operator. Consider the example in Figure 3.19b. Test i is scheduled (by the language processor) to be performed first, and a diagnosis d_1 is output. Usually d_1 is an inquiry-type message to the operator and waits for a response from the operator. If the operator keys in Y (Yes), then test j is immediately executed. Otherwise, when N (No) is keyed in, test k is selected to be performed. More details about operator response are discussed in Section 3.3.2.

3.3 DIAGNOSIS

This section describes the use of diagnosis (see Figure 3.1). The keyword "DIAGNOSIS", label and delimiters (i.e., ":", ";") are similar to those of stimuli discussed in Section 3.2.1. The main diagnosis body can be one of the two syntactical forms: keyword form or positional form. Each field in the keyword form is specified by an English language-like description. While the positional form is a shorthand form of the description, they have a one to one correspondence that will be described in Section 3.3.3.

Syntactically, the diagnosis body consists of two parts: (1) operator messages and (2) operator responses.

TEST 1;

(STIMULI - MEASUREMENTS)

LOGIC: & d_i

TEST 2;

(STIMULI - MEASUREMENTS)

LOGIC: &$\neg$ d_i

TEST 3;

(STIMULI - MEASUREMENTS)

LOGIC: & d_i

(a)

$TEST_i$;

$(STIMULI\_MEASUREMENTS)_i$

LOGIC: *d_1

$TEST_j$;

LOGIC: ?d_1;

TEST k;

LOGIC: ?$\neg d_1$;

(b)

FIGURE 3.19 ILLUSTRATIONS OF LOGIC FUNCTION

3.3.1 OPERATOR MESSAGES

The operator message itself is composed of as many as four parts.

(1) Affected Components. This is a list of failure functions indicating the modes of failure together with the identification symbols of the corresponding components which the diagnosis asserts to have failed. All the affected components can be either conjunctive or disjunctive, but not mixed. For instance, the example in Figure 3.20 indicates that at least one component of R1/R2/R4/R6 is faulty. Its failure function is S (short circuit) which will be defined separately under UUT components/failures in Section 4.4. Note that S(R1/R2/R4/R6) is an abbreviated form of S(R1)/S(R2)/S(R4)/S(R6). Each affected component can be specified either (1) by the failure function followed by the component id enclosed in parentheses, or (2) by the component-failure sequence number which uniquely identifies the affected component. This too is further explained in Section 4.4.

(2) Other Parameters. These will be inserted in the diagnosis message. Each parameter is a variable, a number, or a string constant.

(3) Message Type. This refers to the message identification where a message identification and an optional alias, together with the full text of respective messages must be provided separately (See Section 3.4). Suppose the definition of message type 1 in Figure 3.20 is:

MESSAGE #1: 'AUDIO DISTORTION GREATER THAN (P1) PERCENT.';

Thus, whenever this diagnosis (15) is selected, the above message, with 25% substituting for P1, is the output to the operator.

(4) Timing. The timing specification indicates when the message is to be sent to the operator in respect to the beginning of the application of the stimuli. When no time specification is provided, this implies

```
DIAGNOSIS 15:
OPERATOR MESSAGE:
      AFFECTED COMPONENTS = S(R1 | R2 | R4 | R6),
      OTHER PARAMETERS = (' 25% '),
      TYPE = #1;
```

FIGURE 3.20. EXAMPLE OF A DIAGNOSIS IN KEYWORD FORM

that the message is to be sent upon conclusion of the stimulus-measurement pair. This feature is provided to enhance the facilities for real time interactive exchanges with the operator. Consider the example in Figure 2.21. Whenever this test module is scheduled, diagnosis 4 is issued first before applying any stimuli or performing any measurements. Usually, it is an inteactive diagnosis. Thus, the test procedure is temporarily suspended in order to receive a value or instruction from the operator. The "Time" parameter having a non-zero value assigned to it is logically valid, but it is seldom used.

3.3.2 OPERATOR RESPONSE

When the message contains instructions to the operator to perform duties such as pressing keys, reading meters or making measurements, a response from the operator may be necessary in order to conclude the tests. The specification of required responses consists of one of the following:

(1) Y (YES). The operator instructs to proceed with the suspended test or to initiate a subsequent test.

(2) N (NO). The operator does not want to continue normal testing. In this case the test will be concluded and a connection to subsequent activity may be indicated by use of the logical operator A¬.

(3) VAR. The operator would key into the terminal the values of listed variables (e.g., measured values, selecting options), and key in "Y" to proceed.

```
TEST;
    (STIMULI              )
    (MEASUREMENTS         )
    LOGIC:  *4
    DIAG 4:
        OPERATOR MESSAGE:
            TYPE = #5,
            TIME = 0.00000 E + 00,
        RESPONSE = (VAR1);
```

FIGURE 3.21. EXAMPLE OF THE USAGE OF "TIME" OPTION FOR INTERACTIVE DIAGNOSIS

3.3.3 POSITIONAL DIAGNOSIS

To avoid laborous writing of lengthy keyworded language, a shorthand form called positional diagnosis can be used. Figure 3.22 shows the EBNF specification for this alternative.

For instance, DIAG 29: (, (P1, '%'),D); is equivalent to:

DIAGNOSIS 29:

OPERATOR MESSAGE:

OTHER PARAMETERS = (P1, '%'),

TYPE = D;

and DIAG 24: (,, #15, 0), ? ; is equivalent to:

DIAGNOSIS 24:

OPERATOR MESSAGE:

TYPE = #15

TIME = 0.00000E+00,

RESPONSE = ?;

Note that ? represents "Y/N" for response symbols in current implementation. Also, the commas in operator message clearly divide the message into four subfields, namely (1) affected components, (2) other parameters, (3) message type, and (4) timing, i.e., exactly the same fields as defined before. Missing information between commas indicates that subfield is optional.

3.4 MESSAGE DEFINITION

It has been found that a large number of diagnoses can share a few messages. A few message types are considered to be adequate for several diagnoses since they may be modified by inserting parameters in the indicated locations in the message at execution time when the diagnosis is selected.

```
< POSITIONAL_DIAG > ::=  [ < OPERATOR_MESSAGE > ] [, < OPERATOR_RESPONSE > ]

< OPERATOR_MESSAGE > ::=  ( [ < AFFECTED_COMPONENTS > ]

   [, [ < OTHER_PARAMETERS > ]   [, [ < TYPE > ]   [, < TIMING > ] ] ] ] )

   | < AFFECTED_COMPONENTS >
```

FIGURE 3.22. EBNF SPECIFICATION OF A POSITIONAL DIAGNOSIS

Consequently, a list of message definitions must be provided in the test module specification. The syntax diagram of message definition is depicted in Figure 3.23. Keywords (MESSAGE, ALIAS, TEXT) and label have the usual meanings. Message text is just a character string with some parameters to be inserted during execution time. For an example, see Figure 3.24.

The option of "ALIAS" is just giving a synonymous name to the current message text. We can use either the message 18 or #18 in the test module in the example shown in Figure 3.24.

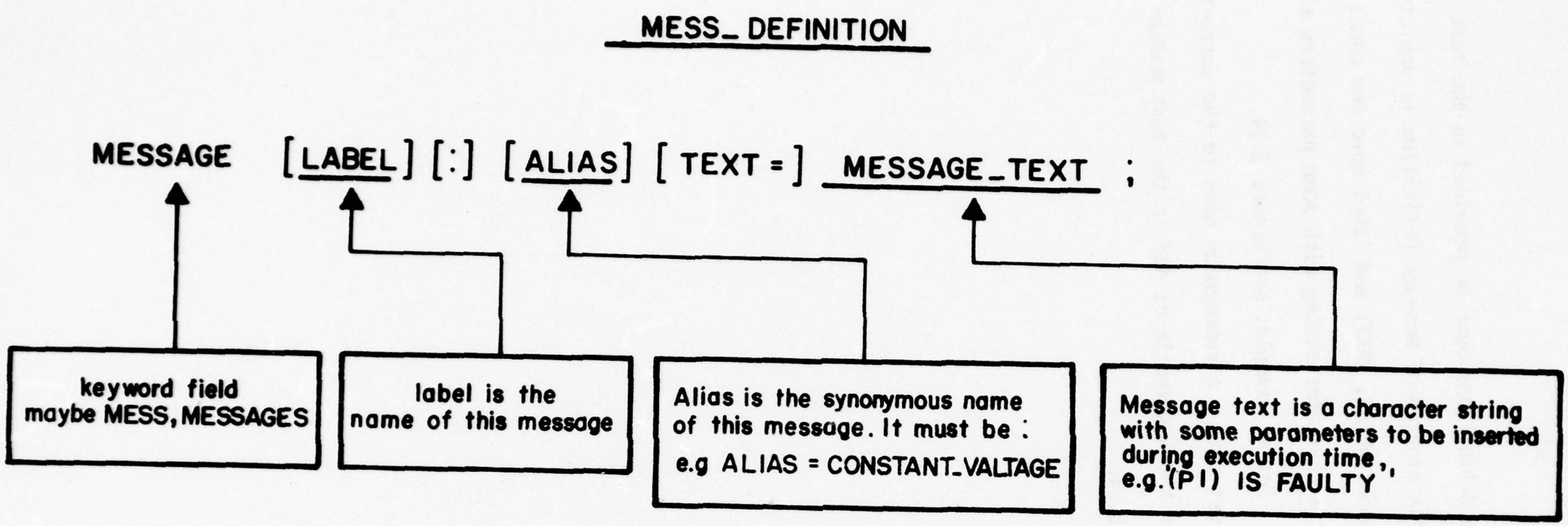


FIGURE 5.23 ILLUSTRATION OF MESSAGE STATEMENTS

```
MESS 18:  ALIAS = #18,
        '(P1) COMPONENT DISTORTION GREATER THAN (P2) PERCENT.';
```

FIGURE 3.24. EXAMPLE OF A MESSAGE STATEMENT

SECTION 4

UUT SPECIFICATION

UUT (or Unit Under Test) related information needed for the automatic program generation is organized into two sections: (1) interconnecting points which are used for identification of connecting points and (2) component failures which identify all the possible faulty components with the failure modes (types of failures).

4.1 UUT CONNECTION POINTS

The general structure of a UUT Connection Point is shown in Figure 4.1. UUT_POINT_ID, a symbolic name, is used to identify connecting points, e.g., J24_B, J1_4, etc. are UUT_POINT_ID. SEQ# which is optional is just an internal sequence number for the NOPAL Processor and has no significant meaning to the user. UUT_KEYWORD is the main body of this section and consists of the following information:

(1) Alias. A synonym to the UUT_POINT_ID. For example,

J1_H, ALIAS = GROUND

(2) Connector. This is a code identifying the type of connector used with the UUT and/or the connecting point on the connector. For example, UUT_PT3: J16, CONNECTOR = (COAX, C);
where coaxial cable is used as a connector and "C" on the coaxial cable is indicated as the connecting point.

(3) Limit. This information is intended to protect the connecting points from inadvertent damage to the UUT caused by excessive stimuli power. The information consists of maximum and minimum

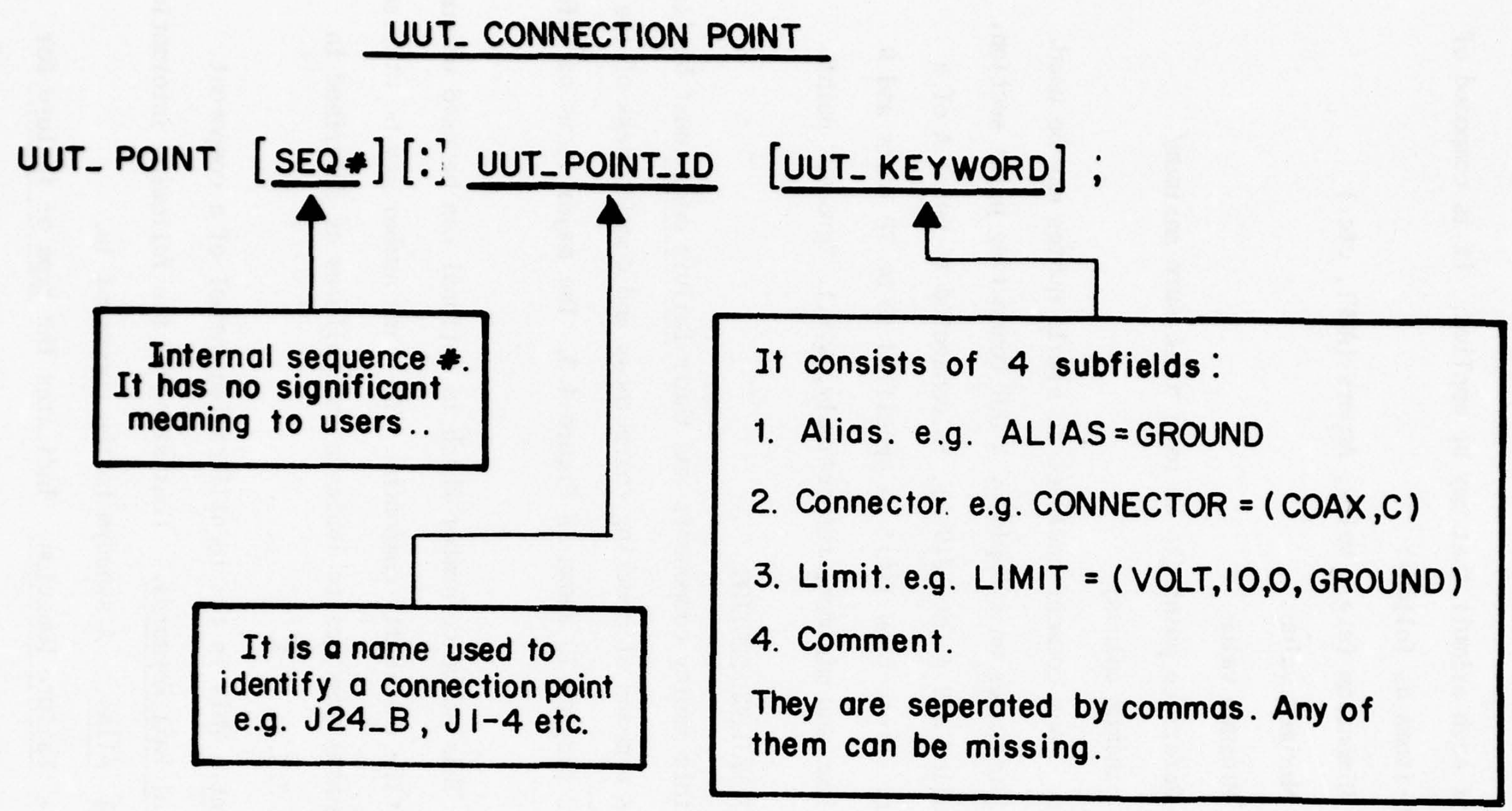


FIGURE 4.1 ILLUSTRATION OF UUT_CONNECTION_POINT STATEMENT

values of each stimuli that may be applied. It is composed of four sub-items as follows:

(a) Dimention (e.g., volts, Ampers (AMP), etc.)

(b) Maximum value

(c) Minimum value

(d) Reference point (to be used to measure maximum/ minimum values)

(4) Comments. Any comment enclosed in signle quotes may be used.

Figure 4.2 illustrates an example of a UUT connecting point section. A UUT point 40, called J19_A (or XJ19_A), is connected to point A of a multiple connector. Protection limit is specified to be 70 volts and 0 volts for the maximum and minimum, respectively, w.r.t. "ground" point.

4.2 UUT COMPONENT FAILURE SECTION

All the possible faulty components and their failure modes must be listed in this section as a means of checking correctness and completeness of the tests. Its' syntax diagram is shown in Figure 4.3. The major items of information are:

(1) SEQ#. This sequence number which is optional can be used uniquely to identify the faulty component. The same number may be utilized in diagnosis messages to indicate the failures as described in Section 3.3.1.

(2) Component. This is the identification symbol of a component.

(3) Component Fail Keywords. Consisting of the following information:

(a) Alias. A synonym to the component id.

(b) Failure Function. Indicates the type of failure for the respective component, e.g., Open, Short, Out of Tolerance, etc.

UUT PT 40: J19_A, ALIAS = XJ19_A, CONNECTOR = (MULTIPLE, A),

LIMIT = (VOLT, 70, 0, GND), 'MULTIPLE CONNECTOR';

FIGURE 4.2. EXAMPLE OF UUT_CONNECTION_POINT STATEMENT

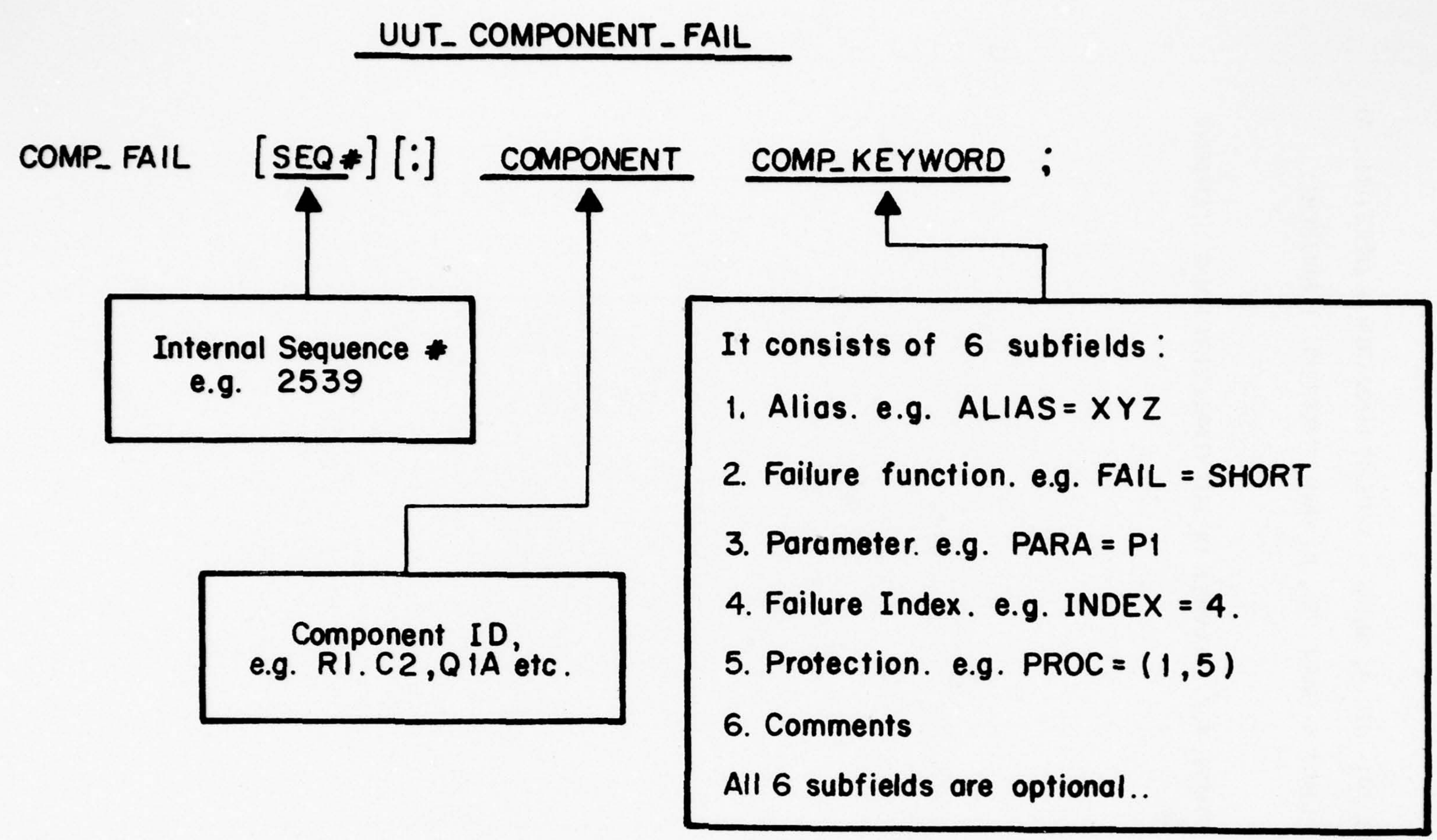


FIGURE 4.5 ILLUSTRATION OF UUT_COMPONENT_FAILURE STATEMENT

(c) Parameters. The names of failure function variables, if any.

(d) Failure Index. This is a number used to index the components by likelihood of failure. The smaller the number, the larger the likelihood. It can be used by the test sequencing program to assign precedence in executing tests for possible failures of components with higher failure likelihoods.

(e) Protection. A list of other component identifications whose failures will prohibit testing a given component.

(f) Comments.

Figure 4.4 illustrates an example of an entry of the UUT component failure specification. STD_5MHZ_FREQ and FREQ_TOL are the names of component and failure function, respectively. Should Component 1 and/or 11 fail, this component could not be tested.

```
COMPONENT_FAILURE 2:
        STD_5MHZ_FREQ, FAIL = FREQ_TOL,
        INDEX = 1, PROT = (1,11):
```

FIGURE 4.4. EXAMPLE OF UUT COMPONENT_FAILURE STATEMENT

SECTION 5

ATE SPECIFICATION

ATE related information needed to verify test module specifications and the UUT specification fall into two parts: (1) ATE connecting points to match connectors of the UUT and (2) functions specified in the stimuli and measurements parts.

5.1 ATE CONNECTION POINT SPECIFICATION

ATE connection point specification is used to describe an adapter used to interface with the connecting points of the UUT matching connector. Its syntax diagram is drawn in Figure 5.1, and an example is shown in Figure 5.2. The example shows ATE point named ATEPT#30 (or H3_A) is connected to UUT points J16 and J22. Note that the UUT points have been previously defined in Section 4.1.

5.2 ATE FUNCTION SPECIFICATION

Functions used for specification stimuli, measurements and for component failures must be stored in the memory of the computer for reference at text execution time. Purely computational functions, such as RANDOM (Generate a random value), 'MAX' (Find the maximum value), etc. and control functions, such as 'REPEAT' (Form a DO-loop control structure), etc. may also be used. The syntax diagram of function specification is shown in Figure 5.3.

Function keywords contain the following items:

(1) Alias. A synonym to the function name.

(2) Type. Specifying the function type. There are five of them: S (Stimuli), M (Measurements), F (Failures), E (Evaluation), and C (Controls).

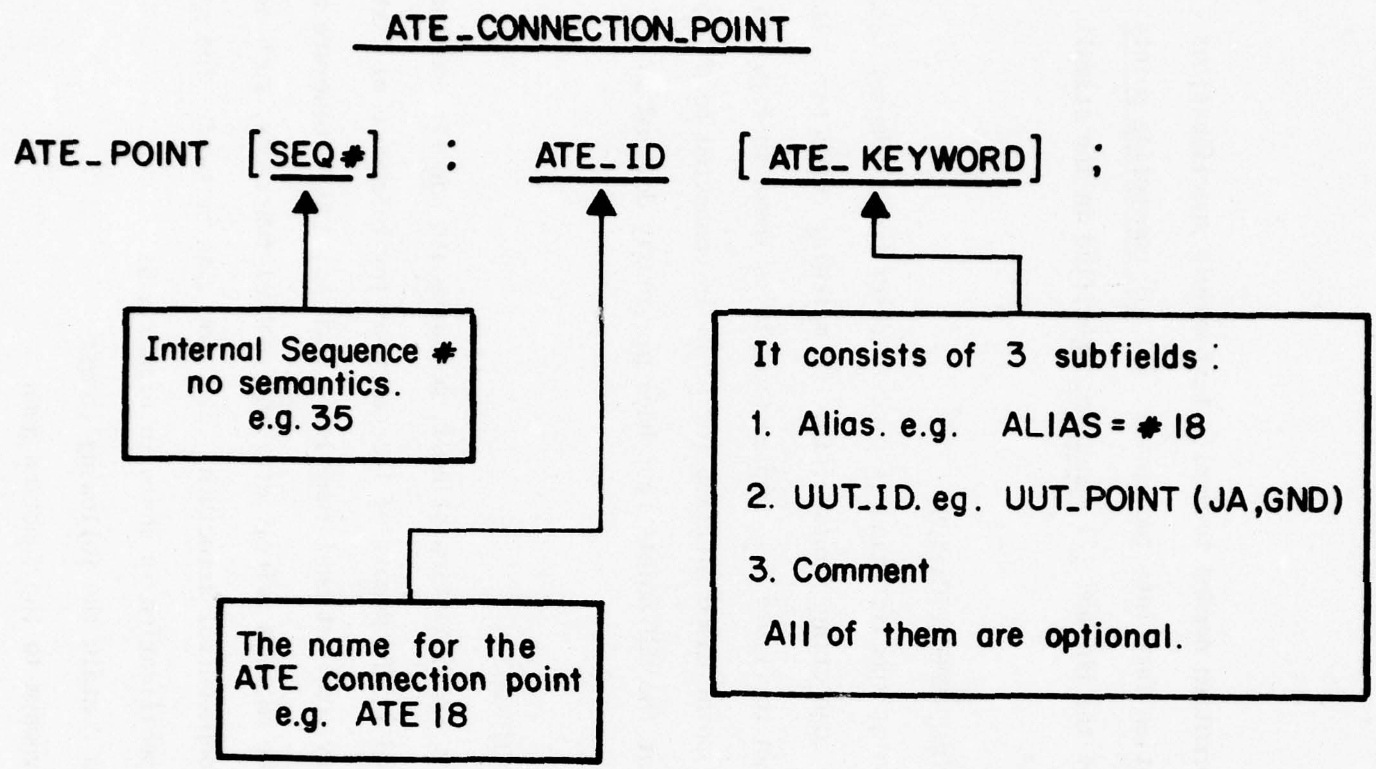


FIGURE 5.1 ILLUSTRATION OF ATE_CONNECTION_POINT STATEMENT

```
ATE_POINT 1:  ATEPT#30, ALIAS = H3_A,
     UUT_PTS = (J16, J22);
```

FIGURE 5.2. EXAMPLE OF ATE CONNECTION POINT STATEMENT

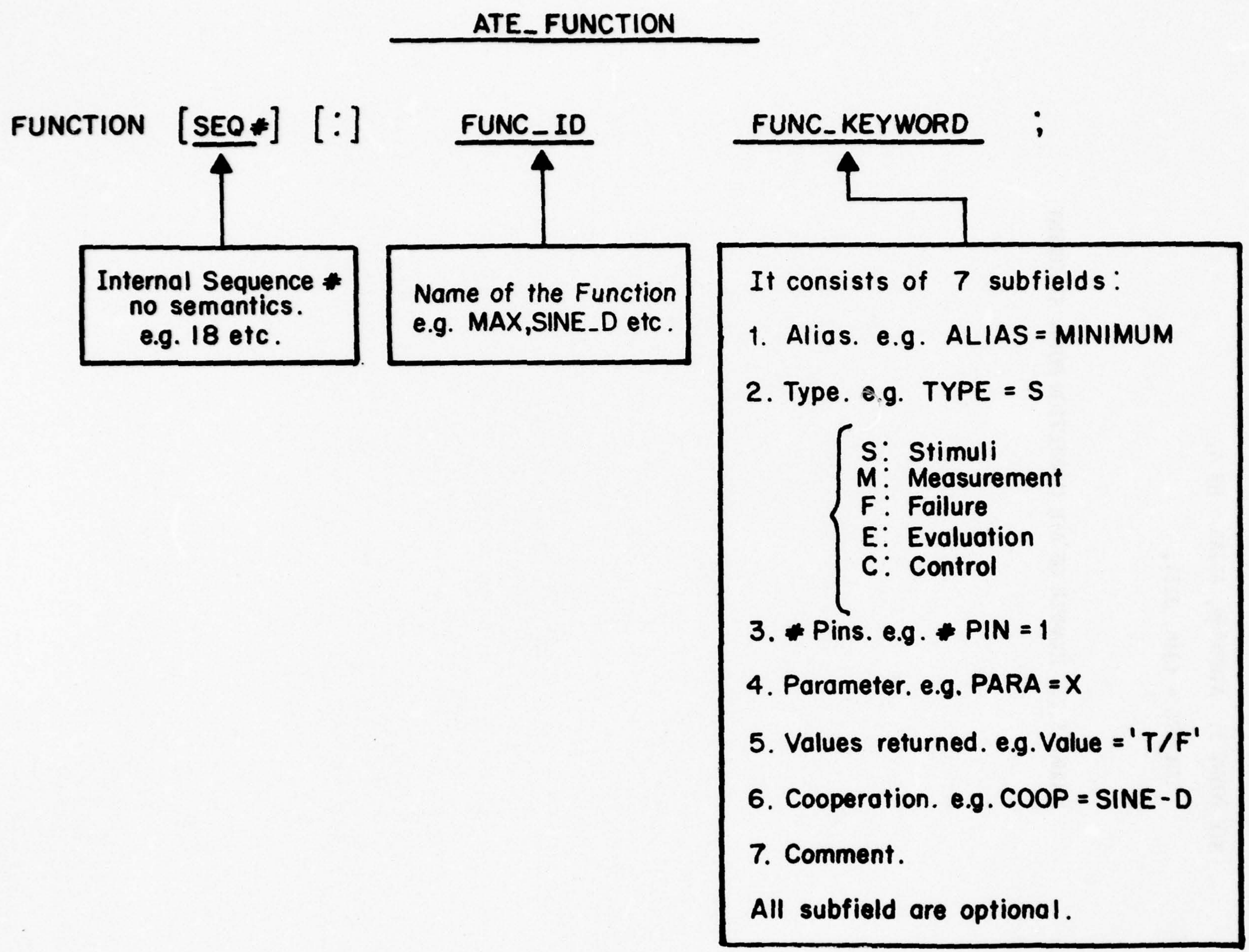


FIGURE 5.3 ILLUSTRATION OF ATE-FUNCTION STATEMENT

(3) #Pins. Indicating the number of pins needed for a function of type S/M.

(4) Parameters. Include parameter name, parameter type, and parameter limits. Parameter limits contain the information of dimension, maximum/minimum values.

(5) Values Returned. By the execution of the called function, the measured or evaluated value(s) may be returned to the calling routine. This item, a character string, provides the description of returned values.

(6) Cooperation. Specifying the functions needed for parallel execution with the given function. This applies especially to the synchronization of stimuli and measurements functions.

(7) Comments.

An example of ATE function entry is shown in Figure 5.4. This function, named "CONST_S" (or "CONSTANT_STIMULI"), is a stimuli function with a formal parameter "X". The range of "X" can be from 60 volts to zero volt. This function generates a constant voltage, namely X volts, as a stimuli to the ATE.

```
FUNC 10:  CONST_S, ALIAS = CONSTANT_STIMULI, TYPE = S,
          PARM = (X, S, LIMIT = (V, 60,0)),
          VALUE RETURNED = 'CONSTANT VOLTAGE.';
```

FIGURE 5.4. EXAMPLE OF ATE FUNCTION STATEMENT

APPENDIX A

```
1 <STRING_CONST>  ::=  <CHAR_STRING>  | <BIT_STRING>                                1
  2 <CHAR_STRING>  ::=  '[<FULL_CHAR>]*'                                            2
    3 <FULL_CHAR>  ::=  <LETTER> | <DIGIT> | <SPECIAL_CHAR>                         3
      4 <LETTER>  ::=  A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z|@|#|$|_  4
      4 <DIGIT>  ::=  0|1|2|3|4|5|6|7|8|9                                           5
      4 <SPECIAL_CHAR>  ::=  .|<|(|+|||&| |$|*|)|;|¬|-|/|,|%|_|>|?|:|#|@|'|=|"|BLANK 6
  2 <BIT_STRING>  ::=  '<BIT> [<BIT>]*'B                                            7
    3 <BIT>  ::=  0|1                                                               8

1 <NUMBER>  ::=  [<SIGN>] <UNSIGNED_NUMBER>                                         9
  2 <SIGN>  ::=  +|-                                                                10
  2 <UNSIGNED_NUMBER>  ::=  <DECIMAL_NUMBER> [<EXPONENT>]                           11
    3 <DECIMAL_NUMBER>  ::=  <UNSIGNED_INTEGER> [<DECIMAL_FRACTION>]                12
                          | <DECIMAL_FRACTION>
      4 <UNSIGNED_INTEGER>  ::=  <DIGIT> [<DIGIT>]*                                 13
      4 <DECIMAL_FRACTION>  ::=  .<UNSIGNED_INTEGER>                                14
    3 <EXPONENT>  ::=  E[<SIGN>] <DIGIT> [<DIGIT>]                                  15

1 <INTEGER>  ::=  [<SIGN>] <UNSIGNED_INTEGER>                                       16

1 <VARIABLE>  ::=  <IDENTIFIER> [(<SUBSCRIPT_LIST>)]                                17
  2 <IDENTIFIER>  ::=  <LETTER> [<TAIL>]*                                           18
```

APPENDIX A EBNF SPECIFICATIONS OF NOPAL

```
3 <TAIL>  ::=  <LETTER> | <DIGIT>                                                        19
  2 <SUBSCRIPT_LIST>  ::=  <ARITH_EXPR> [,<ARITH_EXPR>]*                                 20

1 <ARITH_EXPR>  ::=  [<SIGN>] <TERM> [<ADD_OP> <TERM>]*                                  21
  2 <TERM>  ::=  <FACTOR> [<MULT_OP> <FACTOR>]*                                          22
    3 <FACTOR>  ::=  <PRIMARY> [** <PRIMARY>]*                                           23
      4 <PRIMARY>  ::=  <UNSIGNED_NUMBER> | <VARIABLE> | <FUNCTION_CALL>                 24
                      | (<ARITH_EXPR>)
        5 <FUNCTION_CALL>  ::=  <FUNCTION_ID> [(<ARGUMENT>  ,<ARGUMENT> *)]              25
          6 <FUNCTION_ID>  ::=  <IDENTIFIER>                                             26
          6 <ARGUMENT>  ::=  <ARITH_EXPR> | <STRING_CONST>                               27
    3 <MULT_OP>  ::=  * | /                                                              28
  2 <ADD_OP>  ::=  + | -                                                                 29

1 <COMMENT>  ::=  /* [<FULL_CHAR>]* */                                                   30

1 <IF_CLAUSE>  ::=  IF <BOOLEAN_TERM> THEN                                               31
  2 <BOOLEAN_TERM>  ::=  <BOOLEAN_FACTOR> [V <BOOLEAN_FACTOR>]*                          32
    3 <BOOLEAN_FACTOR>  ::=  <BOOLEAN_PRIMARY> [& <BOOLEAN_PRIMARY>]*                    33
      4 <BOOLEAN_PRIMARY>  ::=  <RELATIONAL_EXPR> | ¬ (<BOOLEAN_TERM>)                   34
        5 <RELATIONAL_EXPR>  ::=  <ARITH_EXPR> <RELATION> <ARITH_EXPR>                   35
          6 <RELATION>  ::=  = | > | < | >= | <= | ¬= | ¬> | ¬<                          36
```

APPENDIX A EBNF SPECIFICATION OF NOPAL (continued)

```
1 <CONN_DIM_EX>  ::=  <CONNECTOR> [<DIMENSION>]                                                    37
  2 <CONNECTOR>  ::=  <CONNECTOR_ID> | < <CONNECTOR_ID> [,<CONNECTOR_ID>]* >                        38
    3 <CONNECTOR_ID>  ::=  <UUT_POINT_ID>                                                           39
  2 <DIMENSION>  ::=  [<PREFIX>] <BASIC_UNIT> [/SEC/SEC] | <TIME_DIMENSION>                         40
    3 <PREFIX> ::= <|G|M|C|U|N|P|MEG                                                                41
    3 <BASIC_UNIT> ::= G|M|CD|CM|DB|FD|FT|GM|IN|HP|HZ|LB|LI|LM|PC|AMP|BAR|DEG|ERG|GAL                42
                  |LUX|OHM|PPM|PSI|RAD|RPH|RPM|RPS|CU_M|DEGC|DEGF|DYNE|INHG|LINE
                  |MMHG|NEWT|SQ_M|STER|VOLT|WATT|CU_FT|FT_LB|HENRY|JOULE|IND_IP
                  |POUNDAL|BRAKE_HP|%
    3 <TIME_DIMENSION>  ::=  [<PREFIX>] <TIME_UNIT>                                                  43
      4 <TIME_UNIT> ::= HR|MIN|SEC                                                                  44

1 <VAL_DIM_EX>  ::=  <ARITH_EXPR>  <DIMENSION>                                                      45

1 <FUNC_DIM_EX>  ::=  <FUNC_TERM> [<ADD_OP> <FUNC_TERM>]*                                           46
  2 <FUNC_TERM>  ::=  <FUNC_FACTOR> [<MULT_OP> <FUNC_FACTOR>]*                                      47
    3 <FUNC_FACTOR>  ::=  [<FUNC_MODIFIER> <MULT_OP>] <FUNC_PRIMARY>                                48
      4 <FUNC_MODIFIER>  ::=  <UNSIGNED_NUMBER>                                                     49
      4 <FUNC_PRIMARY>  ::=  <FUNCTION_ID> [(<FUNC_ARG> [,<FUNC_ARG>]* )]                           50
                  | (<FUNC_DIM_EX>)
        5 <FUNC_ARG>  ::= [<RELATION>] <VAL_DIM_EX> | [=] <RANGE> | <STRING_CONST> | *              51
          6 <RANGE>  ::=  <VAL_DIM_EX>  +- <VAL_DIM_EX> | <VAL_DIM_EX> +- <ARITH_EXPR> [%]           52
```

APPENDIX A EBNF SPECIFICATION OF NOPAL (continued)

```
1 <NOPAL_SPECIFICATION>  ::=  [NOPAL] SPECIFICATION [<SPEC_NAME>] ;                          53
                                   [<NOPAL_STMTS>]*
                               END [<SPEC_NAME>] ;
  2 <SPEC_NAME>  ::=  <LABEL>                                                                 54
    3 <LABEL>  ::=  <IDENTIFIER> | <UNSIGNED_INTEGER>                                         55
  2 <NOPAL_STMTS>  ::=  <TEST_MODULE_SPEC> | <UUT_SPEC> | <ATE_SPEC>                          56
    3 <TEST_MODULE_SPEC>  ::=  <TEST_STEP>                                                    57
                             | <DIAGNOSIS_DEFINITION> [<DIAGNOSIS_DEFINITION>]*
                             | <MESSAGE_DEFINITION> [<MESSAGE_DEFINITION>]*
      4 <TEST_STEP>  ::=  TEST [<TEST_LABEL>] ;                                               58
                        | STIMULI <STIM_ID> [:<BACK_REFERENCE>[<DECLARATION>]* ] ;
                        | MEASUREMENT <MEAS_ID> [ :<BACK_REFERENCE>[<DECLARATION>]*];
                        | LOGIC <LOGIC_ID> [:] <LOGIC_DIAG_LIST> ;
                        | <WAVEFORMS>
        5 <TEST_LABEL>  ::=  <LABEL>                                                          59
        5 <STIM_ID>  ::=   <LABEL>  [(<TEST_LABEL>)]                                          60
        5 <MEAS_ID>  ::=  <STIM_ID>                                                           61
        5 <LOGIC_ID>  ::=  <STIM_ID>                                                          62
        5 <LOGIC_DIAG_LIST>  ::=  <LOGOP_DIAGLBL> [,<LOGOP_DIAGLBL>]*                         63
          6 <LOGOP_DIAGLBL>  ::=  <LOGICAL_OPERATOR> <DIAG_LABEL>                             64
            7 <LOGICAL_OPERATOR>  ::=  <LOGICAL_CONNECTIVE> [<AFTER>] | <AFTER>               65
```

APPENDIX A EBNF SPECIFICATION OF NOPAL (continued)

```
8 <LOGICAL_CONNECTIVE>  ::=  V | & | V¬ | &¬ | *                                        66
8 <AFTER>  ::=  A | A¬                                                                    67
5 <WAVEFORMS>  ::=  <CONJUNCTION> | <ASSERTION> [<ASSERTION>]*                            68
6 <CONJUNCTION>  ::=  CONJUNCTION <WAVEFORM_ID> : <CONJUNCTION_BODY>                      69
                                  [<DECLARATION>]* ;
7 <WAVEFORM_ID>  ::=  [<LABEL>] [(<STIM_MEAS_LABEL>)]                                     70
 8 <STIM_MEAS_LABEL>  ::=  <LABEL>                                                        71
7 <CONJUNCTION_BODY>  ::=  <TRIPLET_CONJUNCT> | <BACK_REFERENCE>                          72
 8 <TRIPLET_CONJUNCT>  ::=  <SIMPLE_CONJUNCTION> | <IF_CONJUNCTION>                       73
  9 <SIMPLE_CONJUNCTION>  ::=  <TRIPLET> [& <TRIPLET>]*                                   74
   10 <TRIPLET>  ::=  (<CONN_DIM_EX> <RELATION> <FUNC_DIM_EX>)                            75
                    | <CONN_DIM_EX> <RELATION> <FUNC_DIM_EX>
  9 <IF_CONJUNCTION>  ::=  <IF_CLAUSE><SIMPLE_CONJUNCTION>[ELSE<TRIPLET_CONJUNCT>]        76
 8 <BACK_REFERENCE>  ::=  [SAME] AS <STIM_MEAS_LABEL>                                     77
                          [EXCEPT <SIMPLE_CONJUNCTION>]
7 <DECLARATION>  ::=  <VARIABLE_TYPE> [:] <VARIABLE_LIST>                                 78
 8 <VARIABLE_TYPE>  ::=  SOURCE | TARGET                                                  79
 8 <VARIABLE_LIST>  ::=  (<VARIABLE> [,<VARIABLE>]*)                                      80
                       | <VARIABLE> [,<VARIABLE>]*
6 <ASSERTION>  ::=  ASSERTION <WAVEFORM_ID> : <ASSERTION_BODY>                            81
                                  [<DECLARATION>]* ;
```

APPENDIX A EBNF SPECIFICATION OF NOPAL (continued)

```
7 <ASSERTION_BODY>  ::=  <SIMPLE_ASSERTION> | <IF_ASSERTION>                                82
  8 <SIMPLE_ASSERTION>  ::=  <RELATIONAL_EXPR>                                             83
                         | <ARITH_EXPR> = <ARITH_EXPR> +- <ARITH_EXPR> [ % ]
  8 <IF_ASSERTION>  ::= <IF_CLAUSE> <SIMPLE_ASSERTION> [ELSE <ASSERTION_BODY>]             84
4 <DIAGNOSIS_DEFINITION>  ::=  DIAGNOSIS <DIAG_LABEL> [:] <DIAG_BODY> ;                    85
 5 <DIAG_LABEL>  ::=  <LABEL>                                                              86
 5 <DIAG_BODY>  ::=  <POSITIONAL_DIAG> | <KEYWORDED_DIAG>                                  87
  6 <POSITIONAL_DIAG>  ::=  [<OPERATOR_MESSAGE>] [,<OPERATOR_RESPONSE>]                    88
   7 <OPERATOR_MESSAGE>  ::=  ([<AFFECTED_COMPONENTS>] [,[<OTHER_PARAMETERS>]              89
                                 [,[<TYPE>]  [,<TIMING>]]]  )
                          | <AFFECTED_COMPONENTS>
    8 <AFFECTED_COMPONENTS>  ::=  <COMPONENT_CONJUNCT> [& <COMPONENT_CONJUNCT>]*           90
                               | <COMPONENT_DISJUNCT> [V <COMPONENT_DISJUNCT>]*
     9 <COMPONENT_CONJUNCT>  ::=  <COMP_FAIL_SEG#> | <COMPONENT>                           91
                              | <FAILURE_FUNCTION> ( <COMPONENT> [& <COMPONENT>]* )
      10 <COMPONENT>  ::=  <IDENTIFIER>                                                    92
     9 <COMPONENT_DISJUNCT>  ::=  <COMP_FAIL_SEQ#> | <COMPONENT>                           93
                              | <FAILURE_FUNCTION> ( <COMPONENT> [V <COMPONENT>]* )
    8 <OTHER_PARAMETERS>  ::=  ( <MSG_ARGUMENT> [,<MSG_ARGUMENT>]* )                       94
                           | <MSG_ARGUMENT>
     9 <MSG_ARGUMENT>  ::=  <STRING_CONST>  |  <VARIABLE> |  <NUMBER>                      95
```

APPENDIX A EBNF SPECIFICATION OF NOPAL (continued)

```
8 <TYPE>  ::=  <MESSAGE_LABEL>                                                           96
8 <TIMING>  ::=  <NUMBER> [<TIME_DIMENSION>]                                             97
7 <OPERATOR_RESPONSE>  ::=  Y/N | (<OP_VAR_LIST>) [,] [Y/N]                              98
                          | <OP_VAR_LIST> [Y/N]
8 <OP_VAR_LIST>  ::=  <VARIABLE> [,<VARIABLE>]*                                          99
6 <KEYWORDED_DIAG>  ::=  [OPERATOR MESSAGE:] <DIAG_KEYWORD> [,<DIAG_KEYWORD>]*          100
7 <DIAG_KEYWORD>  ::=  [AFFECTED] COMPONENT = <AFFECTED_COMPONENTS>                     101
                     | [OTHER] PARAMETER = <OTHER_PARAMETERS>
                     | TYPE = <TYPE>
                     | TIME = <TIMING>
                     | RESPONSE = <OPERATOR_RESPONSE>
4 <MESSAGE_DEFINITION>  ::=  MESSAGE <MESSAGE_LABEL> :                                  102
                     [ALIAS = <SYNONYM>,] [TEXT=] <MESSAGE_TEXT>;
5 <MESSAGE_LABEL>  ::=  <LABEL>                                                         103
5 <SYNONYM>  ::=  <IDENTIFIER>                                                          104
5 <MESSAGE_TEXT>  ::=  <TEXT_ELEM> [[,]<TEXT_ELEM>]*                                    105
6 <TEXT_ELEM>  ::=  <CHAR_STRING>                                                       106
3 <UUT_SPEC>  ::=  <UUT_COMPONENT_FAILURE> [<UUT_COMPONENT_FAILURE>]*                   107
               | <UUT_CONNECTION_POINT> [<UUT_CONNECTION_POINT>]*
4 <UUT_COMPONENT_FAILURE>  ::=  COMP_FAIL[<COMP_FAIL_SEQ#>] [:]<COMPONENT>              108
```

APPENDIX A EBNF SPECIFICATION OF NOPAL (continued)

```
                              [,<COMP_FAIL_KEYWD>]*:
 5 <COMP_FAIL_SEQ#>  ::=  <ENTRY_SEQ#>                                   109
  6 <ENTRY_SEQ#>  ::=  <UNSIGNED_INTEGER>                                110
 5 <COMP_FAIL_KEYWD>  ::=  ALIAS = <SYNONYM>                             111
                     | FAILURE [FUNCTION] = <FAILURE_FUNCTION>
                     | PARAMETER= <PARM_LIST> | INDEX= <FAILURE_INDEX>
                     | PROTECTION= <PROTECTION> | <COMMENTS>
  6 <FAILURE_FUNCTION>  ::=  <FUNCTION_ID>                               112
  6 <PARM_LIST>  ::=  (<PARM_NAME> [, <PARM_NAME>]* ) | <PARM_NAME>      113
   7 <PARM_NAME>  ::=  <IDENTIFIER>                                      114
  6 <FAILURE_INDEX>  ::=  <INTEGER>                                      115
  6 <PROTECTION>  ::= (<COMP_FAIL_ID> [<COMP_FAIL_ID>]*) | <COMP_FAIL_ID>  116
  6 <COMP_FAIL_ID>  ::=  <COMP_FAIL_SEQ#> | <COMPONENT> |                117
                      <FAILURE_FUNCTION>(<COMPONENT>)
  6 <COMMENTS>  ::=  [COMMENT=] <CHAR_STRING>                            118
4 <UUT_CONNECTION_POINT>  ::=  UUT_POINT [<ENTRY_SEQ#>] [:] <UUT_POINT_ID>  119
                              [,<UUT_POINT_KEYWD>]* :
 5 <UUT_POINT_ID>  ::=  <IDENTIFIER>                                     120
 5 <UUT_POINT_KEYWD>  ::=  ALIAS = <SYNONYM>                             121
                     | CONNECTOR = <UUT_CONNECTOR>
                     | LIMIT = <PROTECTIVE_LIMITS> | <COMMENTS>
```

APPENDIX A EBNF SPECIFICATION OF NOPAL (continued)

```
6 <UUT_CONNECTOR>  ::=  (<CONN_TYPE> [, <CONN_POINT>]) | <CONN_TYPE>          122
  7 <CONN_TYPE>  ::=  <IDENTIFIER>                                             123
  7 <CONN_POINT>  ::=  <IDENTIFIER>                                            124
6 <PROTECTIVE_LIMITS>  ::=  ([<DIMENSION>] [, <MAX_LIMIT>] [,[<MIN_LIMIT>]     125
                                        [,<REFERENCE_POINT>]]]  )
                        | <DIMENSION>
  7 <MAX_LIMIT>  ::=  <NUMBER>                                                 126
  7 <MIN_LIMIT>  ::=  <NUMBER>                                                 127
  7 <REFERENCE_POINT>  ::=  <UUT_POINT_ID>                                     128

3 <ATE_SPEC>  ::=  <ATE_FUNCTION> [<ATE_FUNCTION>]*                            129
          | <ATE_CONNECTION_POINT> [<ATE_CONNECTION_POINT>]*
 4 <ATE_FUNCTION>  ::=  FUNCTION [<ENTRY_SEQ#>] [:] <FUNCTION_ID>              130
                                   [,<FUNCTION_KEYWD>]* ;
  5 <FUNCTION_KEYWD>  ::=  ALIAS = <SYNONYM>                                   131
                    | [FUNCTION] TYPE= <FUNCTION_TYPE>
                    | #PINS= <UNSIGNED_INTEGER>
                    | PARAMETER=<PARM> [,PARAMETER=<PARM>]*
                    | VALUE [RETURNED] = <VALUES_RETURNED>
                    | COOPERATION = <COOP_FUNCTIONS> | <COMMENTS>
6 <FUNCTION_TYPE>  ::=  S | M | F | E | C                                      132
```

APPENDIX A EBNF SPECIFICATION OF NOPAL (continued)

```
6 <PARM>  ::=  (<PARM_NAME> [,[<PARM_TYPE>] [,[LIMIT=]<PARM_LIMITS>]] )           133
                  | <PARM_NAME>
  7 <PARM_TYPE>  ::=  S | T                                                       134
  7 <PARM_LIMITS>  ::=  ([<DIMENSION>] [,[<MAX_LIMIT>] [,<MIN_LIMIT>]] )          135
                          | <DIMENSION>
6 <VALUES_RETURNED>  ::=  <CHAR_STRING>                                           136
6 <COOP_FUNCTIONS>  ::=  (<FUNCTION_ID> [, <FUNCTION_ID>]*) | <FUNCTION_ID>        137
4 <ATE_CONNECTION_POINT>  ::=  ATE_POINT [<ENTRY_SEQ#>][:] <ATE_POINT_ID>          138
                                          [,<ATE_POINT_KEYWD>]* ;
 5 <ATE_POINT_ID>  ::=  <IDENTIFIER>                                              139
 5 <ATE_POINT_KEYWD>  ::=  ALIAS = <SYNONYM>                                      140
                        | UUT_POINT = <UUT_POINTS> | <COMMENTS>
6 <UUT_POINTS>  ::=  (<UUT_POINT_ID> [,<UUT_POINT_ID>]* )                          141
                     | <UUT_POINT_ID>
```

APPENDIX A EBNF SPECIFICATION OF NOPAL (continued)

APPENDIX B

APPENDIX B

EXAMPLE: CPS #10559261

The objective of this appendix is to illustrate the use of NOPAL and the output produced at the end of the syntax analysis phase of the NOPAL Processor. Control Power Supply 10559261 is one type of military electronic unit. The appendix consists of four parts: (1) Circuit Diagram, (2) Functional Specification, (3) NOPAL Input Program, and (4) NOPAL Output Reports.

B.1 CIRCUIT DIAGRAM

Figure B.1 shows the circuit diagram of CPS #10559261. This is referred to as a Unit Under Test (UUT). The connector shown on the left of Figure B.1 marked J1, is used for connecting the UUT to a matching connector which interfaces with the Automatic Test Equipment (ATE). At these connecting points, the stimuli may be applied, and measurement can be taken. Definitions of all UUT and ATE connecting points are entered by the user in NOPAL source statement as shown in Table B.2.

B.2 FUNCTIONAL SPECIFICATION

The user usually starts with determination of tests to be performed. In this example, the tests have been provided by the manufacturer and in the corresponding military specification of the circuit. Table B.1 is extracted from Data Sheet # 10559261 (Military MIL-C-14866 (MU), 2 March 1970). This specification of test is further discussed below.

B.3. NOPAL SOURCE SPECIFICATION

Given circuit diagram and functional specification, NOPAL Source Specification is prepared in the following way.

First we will prepare Test Module Specification.

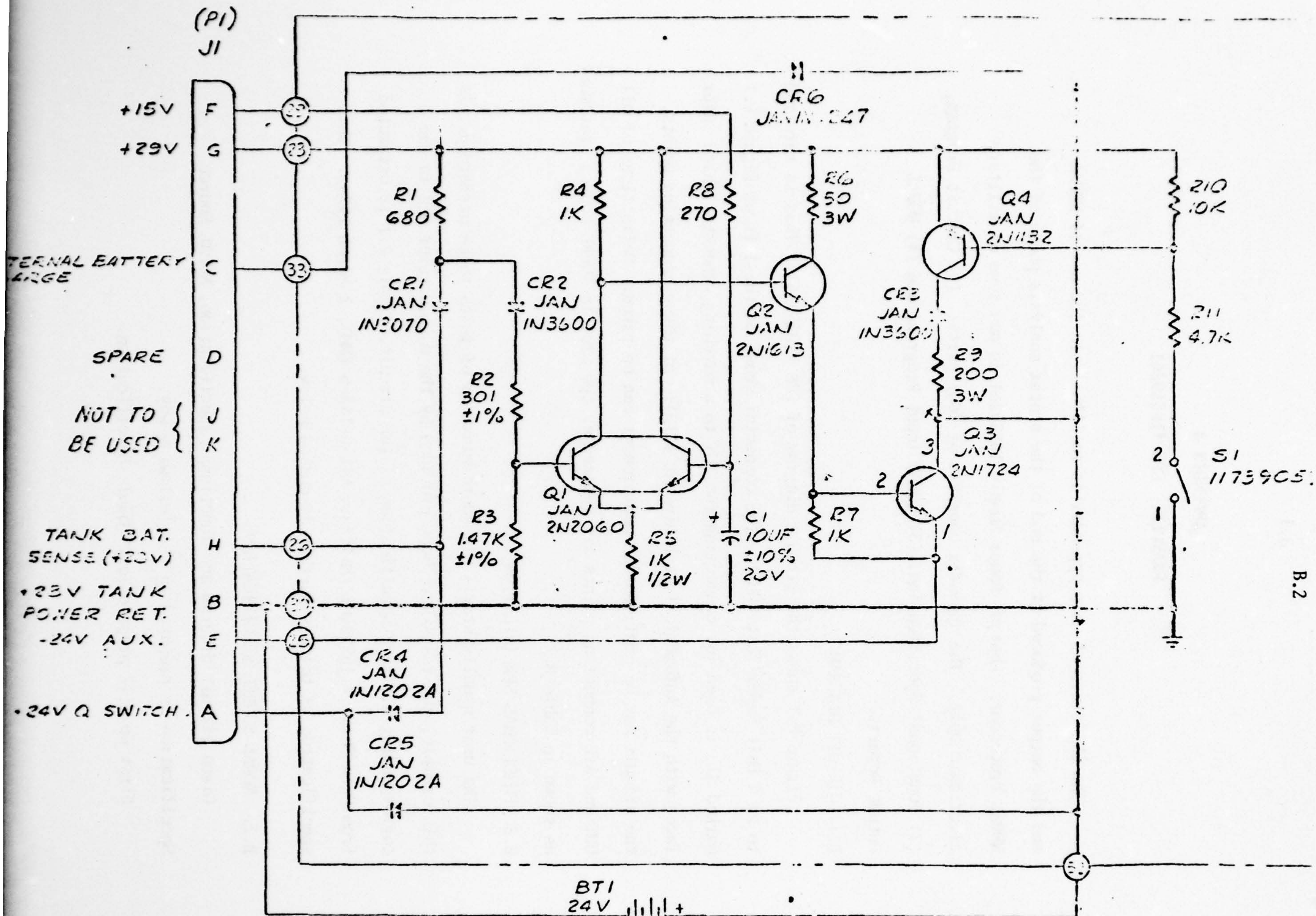


FIGURE B.1 CIRCUIT DIAGRAM OF CPS-10559261

PACKAGING DATA SHEET

10559261 Packaging of Control, Power Supply: 10559261

(Copies of specifications, standards, drawings, and packaging data sheets required by suppliers in connection with specific procurement functions should be obtained from the procuring activity or as directed by the contracting officer.)

3. REQUIREMENTS

3.1 Fabrication. - The power supply control shall be manufactured in accordance with Drawing F10559261 and drawings pertaining thereto.

3.2 General specification. - The power supply control shall meet the following requirements, where applicable, of MIL-F-13926:

(a) Order of precedence
(b) Dimensions and tolerances
(c) Part identification and marking

3.2.1 Electronic workmanship. - The power supply control shall meet the workmanship requirements of Drawing F10559261 and drawings pertaining thereto.

3.3 Performance. - *Unless otherwise specified*, the power supply control shall meet the performance requirements of this specification at standard ambient temperatures between 60 degrees Fahrenheit (^{o}F) and 90^{o}F.

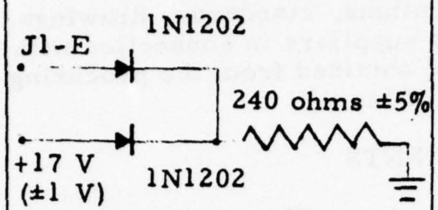
3.3.1 Loads, power and signals. - The power supply control shall perform as specified herein when the loads, power and signals of Table I are applied as specified.

TABLE I

ITEM	CONDITION	CHARACTERISTICS	CONNECTIONS
1.	Loads:		As specified herein
1.1	Resistor	2.2 kilohms, ±5%	

TABLE B.1 FUNCTIONAL SPECIFICATION OF CPS-10559261

TABLE I (continued)

ITEM	CONDITION	CHARACTERISTICS	CONNECTIONS
1. 2	Diode-resistor network		
2.	Power sources:	Tolerance / Maximum p-p ripple	Connected between following pins of J1:
2. 1	15 Vdc	±0. 7 Vdc 25 mV	F(+) and B(-)
2. 2	29 Vdc	±1 Vdc 25 mV	G(+) and B(-)
2. 3	17 Vdc	±1 Vdc 25 mV	(See item 1. 2 above)
3.	Signal sources:		
3. 1	Analog	±14 Vdc to +32 Vdc, adjustable	As specified herein.

3. 3. 1. 1 Battery charging current.

3. 3. 1. 1. 1 With the signal voltage of Table I adjusted to 24 ±1 Vdc and applied between J1-H(+) and J1-B(-), and the 2. 2-kilohm load of Table I connected between J1-A and J1-B, the dc current at J1-G shall be 47 ±10 mA.

3. 3. 1. 1. 2 Under the conditions of 3. 3. 1. 1. 1, a reduction of the temperature of thermostat S1 to less than 14°F shall cause the current at J1-G to decrease by at least 10 mA.

3. 3. 1. 2 Q-switch voltage.

3. 3. 1. 2. 1 Under the conditions of 3. 3. 1. 1. 1, but with the signal voltage of Table I adjusted to 31 ±1 Vdc and applied between J1-H(+) and J1-B(-), the voltage at J1-A shall be 30. 5 ±1. 5 Vdc.

3. 3. 1. 2. 2 Under the conditions of 3. 3. 1. 2.1, but with the signal voltage of Table I adjusted to 15 ±0. 5 Vdc and applied between J1-H(+) and J1-B(-), the voltage at J1-A shall be 23 ±6 Vdc.

TABLE B.1 FUNCTIONAL SPECIFICATION OF CPS-10559261 (continued)

3. 3. 1. 3 +24-volt output.

3. 3. 1. 3. 1 With the diode-resistor load of Table I connected as specified, and with the signal voltage of Table I adjusted to 24 ±1 Vdc and applied between J1-H(+) and J1-B(-), the voltage across the 240-ohm load shall be 16 ±2 Vdc.

3. 3. 1. 3. 2 Under the conditions of 3. 3. 1. 3. 1, but with the voltage at J1-H adjusted to 15 ±0. 5 Vdc, the voltages across the 240-ohm load and at pin J1-A shall be within 2 volts.

3. 3. 2 Environmental.

3. 3. 2. 1 Storage temperature. - The power supply control shall meet the requirements of 3. 3. 1 at ambient temperature (60°F to 90°F) after having been exposed to and thermally stabilized at -80°F and +160°F.

3. 3. 2. 2 Operating temperatures. - The power supply control shall meet the requirements of 3. 3. 1. 1. 2, 3. 3. 1. 2 and 3. 3. 1. 3 while exposed to and thermally stabilized at -40°F; and shall meet the requirements of 3. 3. 1. 1. 1, 3. 3.1. 2 and 3. 3. 1. 3 while exposed to and thermally stabilized at +125°F, subsequent to which it shall meet the requirements of 3. 3. 1. 1, 3. 3. 1. 2, and 3. 3. 1. 3 at ambient temperature (60°F to 90°F).

3. 3. 2. 3 Vibration. - The power supply control shall meet the requirements of 3. 3. 1 after being exposed to the following vibratory conditions:

(a) Amplitude:	0. 5 inch maximum
(b) Acceleration:	4g maximum
(c) Crossover frequency:	12. 5 Hertz (Hz)
(d) Sweep time:	5 to 500 to 5 Hz in 15 minutes. Two sweeps in each of the three mutually perpendicular axes.

3. 3. 2. 4 Shock. - The power supply control shall meet the requirements of 3. 3. 1 after being exposed to a total of 6 shock impulses, one in each direction of the three mutually perpendicular axes. Each shock impulse shall be a sawtooth or half-sine wave with a time duration of 6 milliseconds. The peak amplitude of each shock impulse shall be 100g.

3. 3. 2. 5 Immersion. - With the power supply control pressurized to 5 ±1 psig with nitrogen or air, and submerged no more than 2 feet in water at a temperature of 110 ±10°F, there shall be no evidence of gas bubbles emanating from inside the unit during a 15-minute observation period.

TABLE B.1 FUNCTIONAL SPECIFICATION OF CPS-10559261 (continued)

To specify the first test, refer to paragraph 3.3.1.1.1, "Testing battery charging current" Table B.1. The test module specification is shown in lines 5 to 11, Table B.2. First assign a test name for the test, e.g., 55111. Next assign a name to the stimuli part, e.g., S55111. If they are not supplied, the NOPAL Processor will automatically assign uniques names for them. The functional specification (see Table B.1, 3.1.1.1.1) says: "With the signal voltage of Table 1 adjusted to 24 ± 1 VDC and applied between J1_H (+) and J1_B (-), and the 2.2 Kilohm load of Table 1 connected between J1_A and J1_B,...." It implies the following stimuli conjunctions:

(1) Between G and B --- Apply 29 VDC. 1000 mA is obtained by roughly calculating the maximum current between points G and B. (Refer to item 2.2 of Table B.1)

(2) Between F and B --- Apply 15 VDC. Maximum current will be 1 mA. (Refer to item 2.1 of Table B.1)

(3) Between H and B --- Apply 24 VDC. Maximum current 100 mA. (Refer to specified requirement in 3.3.1.1.1)

(4) Between A and B --- Connect a 2200 ohm load (Refer to specified requirement in 3.1.1.1.1).

Therefore, the stimuli of this test (TEST 55111) appears in NOPAL form as:

```
STIMULI S55111;
        CONJUNCTION: <J1_G, J1_B> VOLT = P_SUPPLY (29VOLT, 1000 MAMP) &
                     <J1_F, J1_B> VOLT = P_SUPPLY (15VOLT,    1 MAMP) &
                     <J1_H, J1_B> VOLT = P_SUPPLY (24VOLT,  100 MAMP) &
                     <J1_A, J1_B>      = LOAD_L (2200 OHM);
```

The measurement is taken at point G according to the functional specification (3.3.1.1.1) --- "the DC current at J1_G shall be 47+10mA." Hence, the measurement conjunction is formed to take the DC current at point G. An assertion is made consequently to decide if the measured value (i.e., IE29V) falls within 47+10 mA. The NOPAL measurement statement is then:

```
MEASUREMENT;
        CONJUNCTION:  J1_G MA = AMP_M (IE29V MA)
              TARGET = IE29V;
        ASSERTION:  ABS (IE29V-47) <= 10
        SOURCE :  IE29V;
```

Next part in the test is a LOGIC statement. After calculating the circuit diagram, at least one of R1/R11/CR3/Q1B is faulty if this test fails. Hence, diagnosis 2, with affected components such as R1/R11/CR3/Q1B and message (type 1), is selected to indicate the fault. Diagnosis and message definitions should be provided separately. They are shown in Table B.2 line 132. Note that the non-proceduralness ensures that they can be put on any place of the program in any order. For example, diagnosis 2 looks like:

```
DIAG 2:  AFFECTED COMP = S(R1)/S(R11)/S(CR3)/NS(Q1B),
            TYPE = 1.
```

The message type is a text of character string which may be modifiable by inserting parameters in indicated locations. For example, message type 1 looks like:

```
MESS 1:  'FAULTY COMPONENTS:  (C)';
```

However, the decision relating to fault isolation sometimes cannot be made after a single test, but only after several tests. For instance, by calculation or computer simulation, component Q1A is faulty when all three tests

AD-A035 610 MOORE SCHOOL OF ELECTRICAL ENGINEERING PHILADELPHIA P--ETC F/G 9/2
THE NOPAL LANGUAGE SPECIFICATION AND USER MANUAL.(U)
AUG 76 H CHE, Y CHANG DAAA25-75-C-0650
UNCLASSIFIED 76-04 NL

2 OF 2
ADA035610

END
DATE
FILMED
3-77

(Tests 55111, 55131, 55132) fail. The LOGIC part of above three tests must then include an entry &¬12, where diagnosis 12 will indicate Q1A is faulty. The complete LOGIC part for test 55111 is:

LOGIC 1¬2, &¬8, &¬10, &¬12;

Included is a complete test module specification for one function test requirement in paragraph 3.3.1.1.1 of Table B.1. Similarly, all the other test modules can be specified.*

After finishing test module specification, the UUT and ATE specifications must also be prepared. UUT related information needed for automatic program generation includes two parts: (1) interconnecting points which are used for identifications of connecting points, e.g., UUT_P J1_B, ALIAS = TANK_POWER_RETURN, COMMENT = 'GROUND'; as shown in line 137, Table B.2. Note that all the UUT connection points should be defined, and (2) component failures which identify all the possible faulty components with the type of failures e.g. COMP_FAIL 2: R1, FAIL FUNCTION = S; which is shown in line 51, Table B.2.

ATE related information needed to verify test modules specifications and the UUT specifications falls into two parts: (1) all ATE connecting points to match connectors of the UUT, e.g., ATE_P 30: ATEPT#30, UUT_PTS = (J16, J22). This part of the specification may be omitted if no connectors between UUT and ATE are required.and (2) all ATE functions specified in the stimuli and measurements sections, e.g.,

*This step involving the preparation of the NOPAL source may be automated in the near future. Some progress on that has been reported in "Automatic Test Program Generation for Automatic Testing Systems," written by Cihan Tinaztepe and others in Moore School Technical Report, University of Pennsylvania, March 23, 1976.

FUNC SO, TYPE = F, PARM = COMP; which is shown in line 160, Table B.2. Observe that the UUT and ATE specification are used to provide some supporting information to the test modules. Should any inconsistency occur, it will be caught by the NOPAL Processor. A corresponding error/warning message is printed out to the user.

```
STMT NO.
--------

  1     SPEC CPS#10559261;
  2       TEST;   /* OPERATOR PREPARATION */
  3         LOGIC:    *1;
  4         DIAGNOSIS 1:  TYPE=X;
  5       TEST 55111;
  6         STIMULI S55111;
  7           CONJUNCTION: <J1_G, J1_B> VOLT = P_SUPPLY(29 VOLT, 1000 MAMP) &
  7                        <J1_F, J1_B> VOLT = P_SUPPLY(15 VOLT, 1 MAMP) &
  7                        <J1_H, J1_B> VOLT = P_SUPPLY(24, 100MAMP) &
  7                        <J1_A, J1_B>   =LOAD_L(2200 OHM);
  8         MEASUREMENT;
  9           CONJUNCTION: J1_G MAMP = AMP_M(IE29V MAMP)
  9                          TARGET: IE29V;
 10           ASSERTION:  ABS(IE29V - 47) <= 10      SOURCE:  IE29V;
 11         LOGIC:  !¬2, &¬8, &¬10, &¬12;
 12       TEST 55112;
 13         STIM;
 14           CONJ: SAME AS S55111 EXCEPT  S1 = COOL(14 DEGF);
 15         MEAS;
 16           CONJ:  J1_G = AMP_M(IE29VL MAMP)
 16                       TARGET: IE29VL;
 17           ASRT:   IE29V - IE29VL > 10;
 18         LOGIC: !¬3, &¬8;
 19       TEST 55121;
 20         STIM:  SAME AS S55111 EXCEPT
 20              <J1_H,J1_B> VOLT = P_SUPPLY(31VOLT, 10MAMP);
        /* NOTE: LAST STMT IS EQUIVALENT TO THE FOLLOWING TWO STMTS:
          STIM:   CONJ: SAME AS S55111 EXCEPT <J1_H,J1_B> V = P_SUPPLY(31V, 10MA);    */
 21         MEAS;
 22           CONJ: <J1_A, J1_B> = VOLT_M(VNJ1A VOLT)   TARG:  VNJ1A;
 23           ASSERT:  ABS(VNJ1A-30.5) <= 1.5;
 24         LOGIC:  !¬4;
 25       TEST 55122;
 26         STIM:  CONJ: SAME AS S55111 EXCEPT
 27                      <J1_H,J1_B> = P_SUPPLY(15VOLT, 10MAMP);
```

TABLE B.2. NOPAL SOURCE PROGRAM

```
29    MEAS:
29      CONJ: <J1_A, J1_B> VOLT = VOLT_M( VNJ1A VOLT)    TARG:  VNJ1A;
30      ASRT:  ABS( VNJ1A - 23) < 6;
31    LOGIC  |¬5;
32  TEST 55131;
33    STIM S55131:      SAME AS S55111 EXCEPT
33       <J1_H, J1_B> VOLT = P_SUPPLY(24VOLT, 10MAMP)
33       & <J1_E, J1_B> = LOAD_NL;
34    MEAS:
35      CONJ: <J1_240, J1_B> VOLT = VOLT_M(VR240 VOLT)    TARGET:  VR240;
36      ASSERT: ABS(VR240-16) <= 2;
37    LOGIC  |¬6, &¬9,&¬10,&¬12:
38  TEST 55132;
39    STIM;
40      CONJ:  SAME AS S55131  EXCEPT <J1_G,J1_B> =P_SUPPLY(15VOLT,10MAMP);
41    MEAS;
42      CONJ:  <J1_A,J1_B> = VOLT_M(VR240 VOLT)  &
42             <J1_2200, J1_B>VOLT=VOLT_M(VR2200 VOLT)
42              TARGETS:  VR240, VR2200;

43      ASSERT:  ABS(VR240 - VR2200) <= 2;
44    LOGIC:  |¬7, &¬9,&¬11, &¬12;
45  DIAGNOSIS 2:  AFFECTED COMPONENTS= S(R1 | R11 | CR3) | NS(Q1B), TYPE= 1;
46  DIAG 8: TYPE =1,
46         AFFECTED COMP= O(R9|R11) | S(R9|R10) | ON(Q4) | OO(Q4) | OS(Q4) |
46            NS(Q4) | SN(Q4) | SO(Q4);
47  DIAG 10:COMP= O(R1|R2|R5|CR2) | S(R3|R4) | ON(Q1A) | OO(Q1A) |OS(Q1A)
47         | NS(Q1A) | NO(Q1A) | SS(Q1B|Q2) | SN(Q1B|Q2),
47         TYPE=1;
48  DIAG 12:  TYPE=1, COMP=SO(Q1A);
49  COMPONENT_FAILURE 1: R1, FAILURE FUNCTION=O;
50  COMP_FAIL 2: R1, FAIL FUNC=S;
51  COMPFAIL  3: R2, FAIL=O;
52  COMP 4: R2, FAIL=S;
53  COMP_FAIL 5 R3, FAIL=O;
54  COMP_FAIL 6 R3, FAIL=S;
55  COMP_FAIL 7 R4, FAIL=O;
56  COMP_FAIL 8 R4, FAIL=S;
57  COMP_FAIL  9: R5, FAIL=O;
58  COMP_FAIL 10: R5, FAIL=S;
```

TABLE B.2 NOPAL SOURCE PROGRAM (continued)

```
COMP_FAIL 11: R6, FAIL=O;
COMP_FAIL 12: R6, FAIL=S;
COMP_FAIL 13: R7, FAIL=O;
COMP_FAIL 14: R7, FAIL=S;
COMP_FAIL 16: R8, FAIL=S;
COMP_FAIL 15: R8, FAIL=O;
COMP_FAIL 17: R9, FAIL=O;
COMP_FAIL 18: R9, FAIL=S;
COMP_FAIL 20: R10, FAIL=S;
COMP_FAIL 19: R10, FAIL=O;
COMP_FAIL 21: R11, FAIL=O;
COMP_FAIL 22: R11, FAIL=S;
COMP_FAIL 23: C1, FAIL=O;
COMP_FAIL 24: C1, FAIL=S;
COMP_FAIL 25: CR1, FAIL=O;
COMP_FAIL 26: CR1, FAIL=S;
COMP_FAIL 28: CR2, FAIL=S;
COMP_FAIL 27: CR2, FAIL=O;
COMP_FAIL 29: CR3, FAIL=O;
COMP_FAIL 30: CR3, FAIL=S;
COMP_FAIL 32: CR4, FAIL=S;
COMP_FAIL 31: CR4, FAIL=O;
COMP_FAIL 33: CR5, FAIL=O;
COMP_FAIL 34: CR5, FAIL=S;
COMP_FAIL   : Q1A, FAIL=ON;
COMP_FAIL   : Q1A, FAIL=OO;
COMP_FAIL   : Q1A, FAIL=OS;
COMP_FAIL   : Q1A, FAIL=NS;
COMP_FAIL   : Q1A, FAIL=SS;
COMP_FAIL   : Q1A, FAIL=SN;
COMP_FAIL   : Q1A, FAIL=SO;
COMP_FAIL   : Q1A, FAIL=NO;
COMP_FAIL   : Q1B, FAIL=ON;
COMP_FAIL   : Q1B, FAIL=OO;
COMP_FAIL   : Q1B, FAIL=OS;
COMP_FAIL   : Q1B, FAIL=NS;
COMP_FAIL   : Q1B, FAIL=SS;
COMP_FAIL   : Q1B, FAIL=SN;
COMP_FAIL   : Q1B, FAIL=SO;
```

TABLE B.2 NOPAL SOURCE PROGRAM (continued)

```
COMP_FAIL    : Q1B, FAIL=NO;
COMP_FAIL    : Q2,  FAIL=ON;
COMP_FAIL    : Q2,  FAIL=OO;
COMP_FAIL    : Q2,  FAIL=OS;
COMP_FAIL      Q2,  FAIL=NS;
COMP_FAIL      Q2,  FAIL=SS;
COMP_FAIL      Q2,  FAIL=SN;
COMP_FAIL      Q2,  FAIL=SO;
COMP_FAIL      Q2,  FAIL=NO;
COMP_FAIL    : Q3,  FAIL=ON;
COMP_FAIL    : Q3,  FAIL=OO;
COMP_FAIL    : Q3,  FAIL=OS;
COMP_FAIL      Q3,  FAIL=NS;
COMP_FAIL      Q3,  FAIL=SS;
COMP_FAIL      Q3,  FAIL=SN;
COMP_FAIL      Q3,  FAIL=SO;
COMP_FAIL      Q3,  FAIL=NO;
COMP_FAIL    : Q4,  FAIL=ON;
COMP_FAIL    : Q4,  FAIL=OO;
COMP_FAIL      Q4,  FAIL=OS;
COMP_FAIL      Q4,  FAIL=NS;
COMP_FAIL      Q4,  FAIL=SS;
COMP_FAIL      Q4,  FAIL=SN;
COMP_FAIL      Q4,  FAIL=SO;
COMP_FAIL      Q4,  FAIL=NO;
DIAG 11: COMPS=4 | 10 | SO(Q2), TYPE = 1;
DIAG  3: TYPE=1, COMP=O(CR3) | SS(Q4) | NO(Q4);
DIAG  4: TYPE=1, COMP=O(CR4);
DIAG  5  TYPE=1, COMP=O(CR5);
DIAG  6: TYPE=1, COMPONETS= SS(Q3) | SN(Q3);
DIAG  9: TYPE=1, COMPONET= O(R3);
DIAG  7: TYPE=1, COMP=5 | 15 | 25 | 24 | SS(Q1A) | SN(Q1A) | NO(Q1B|Q2|Q3)
                 | OO(Q2|Q3) | SO(Q3);

MESSAGE X: 'SET UP AS REQUIRED IN CPS#10559261.';
MESS 1: 'FAULTY COMPONENTS: (C)';
```

TABLE B.2 NOPAL SOURCE PROGRAM (continued)

```
132    UUT_POINT 1: J1_F, ALIAS= E15V, LIMIT=(VOLT,15.0, J1_B), COMMENT=
132                 '15V BIASING REF.';
133    UUT_POINT  : J1_G,ALIAS=E29V, LIMIT=(VOLT,15.0, J1_B), COMM='29V MAIN POWER';
134    UUT_P J1_C,ALIAS=EXT_BATTERY_CHARGE, LIMIT=(VOLT,24.0,J1_B),
134                'USE TO REPLACE BT1 OR TO CHARGE BT1';
135    UUT_P    J1_H,ALIAS=TANK_BATTERY_SENSE,LIMIT=(VOLT,31.0,J1_B),COMM=
135              'SENSE SIGNAL FROM MAIN POWER SUPPLY.';
136    UUT_P    J1_B,ALIAS=TANK_POWER_RETURN, COMMENT='GROUND';
137    UUT_P    J1_E,ALIAS=AUX_POWER, LIMIT=(VOLT,31.0,J1_B),
137       'IF VNJ1G=29V & VNJ1H=30V THEN 15V; ELSE 30V.';
138    UUT_PIN:  J1_A,ALIAS=Q_SWITCH, LIMIT=(VOLT,31.0,J1_B),
138            '24V WHEN SENSE<24; ELSE SENSE.';
139    UUT_P    S1, CONNECTOR=AIR_OR_CONTACT,LIMIT=(DEGF,75,14),'COLD AIR ON EQ.';
140    UUT_POINT : J1_2200,ALIAS=LOADRESISTOR;
141    UUT_POINT  : J1_240, COMM='DUMMY POINT';

142    FUNCTION 10: VOLT_M, ALIAS=VOLTMETER, FUNCTION TYPE=M,
142                 PARAMETER=(V,T,(VOLT,31,-31)),VALUE RETURNED='CONST VOLT.',
142                   COMMENT='V-READING';
143    FUNC :  AMP_M,ALIAS=AMPEREMETER,TYPE=M, PARM=(I, T,(AMP,5,-5)),VALUE=
143              'CNST. CURRENT', #PINS=1, COMM='+ READING I INTO UUT';
144    FUNC:   COOL,ALIAS=COOLANT,TYPE=S, PARM=(DESIR,S,LIMIT=(DEGF,76,15)),
144              #PINS=1, 'COOL UNIT UNTIL DESIRED TEMPERATURE.';
145    FUNC:  P_SUPPLY,ALIAS=DC_POWER_SUPPLY, TYPE=S, PARM1=(V,S,LIMIT=(VOLT,34,0)),
145            PARM2=(I,S,LIMIT=( AMP ,2,0));
146    FUNC  LOAD_L,ALIAS=LOAD_LINEAR, TYPE=S, PARM=(R,S,(KOHM,10,0));
147    FUNC   LOAD_NL,ALIAS=NONLINEAR,TYPE=S;
148    FUNCTION ABS,ALIAS=ABSOLUTE, TYPE=E, PARM=(EXP,S), 'TAKE ABS. VALUE OF EXP';
149    FUNCTION :  O,TYPE=F,PARM=(COMPONENT);
150    FUNC  S, TYPE=F, PARM=COMP;
151    FUNC  OV, TYPE=F, PARM=COMP;
152    FUNC  OO, TYPE=F, PARM=COMP;
153    FUNC  OS, TYPE=F, PARM=COMP;
154    FUNC  VS, TYPE=F, PARM=COMP;
155    FUNC  SS, TYPE=F, PARM=COMP;
156    FUNC  SV, TYPE=F, PARM=COMP;
157    FUNC  SO, TYPE=F, PARM=COMP;
158    FUNC  VO, TYPE=F, PARM=COMP;
159  END CPS#10559261;
```

TABLE B.2 NOPAL SOURCE PROGRAM (continued)

B.4 NOPAL OUTPUT

The NOPAL Processor will take the NOPAL prepared by the user and produce the following outputs:

(1) NOPAL Test Specification Summary Report. This is the formatted listing of test specification. All the labels, including the one supplied by users as well as those generated by NOPAL Processor, are shown. Missing items are assigned to their default values. Abbreviations are spelled out. For instance, if a test module does not contain a stimuli, a comment "NULL STIMULI" in the formatted listing indicates the fact. Table B.3, B.4, and B.5 illustrate the formatted listings of test module specification, UUT specification, and ATE specification, respectively.

(2) Error/Warning Messages Generated During NOPAL Syntax Analysis. All the errors, e.g., illegal identifiers, invalid keywords, wrong syntactical structure, etc. are listed in Table B.6.

(3) Cross Reference and Attribute Report. From the report, users can check each occurrence of variables, functions, and connectors, etc. Their declarations, e.g., the synonym of a function, the failure type of a component, etc. are also illustrated in Table B.7.

(4) Error/Warning Messages Generated During Cross Reference. Some semantic inconsistencies, e.g., cyclical back references, ambiguous SOURCE_TARGET relationships and undefined component, etc. are illustrated in Table B.8.

(5) Cross Reference of Diagnosis to Test-Modules. For each diagnosis in Table B.9, the user can check if test modules utilize the correct diagnoses.

(6) Cross Reference Messages to Diagnoses to Tests. Each entry indicates which diagnoses/tests refer to a particular message. (See Table B.10)

(7) Cross Reference of Affected Components to Diagnoses to Tests (See Table B.11)

(8) Cross Reference of UUT Connection Point to Tests to ATE Connection Point. (See Table B.12)

(9) Cross Reference of ATE Function to Test Modules. (See Table B.13)

```
/* NOPAL TEST SPECIFICATION SUMMARY REPORT, FILE: SOURCE2 */


   /**************************************************/
   /*                                                */
   /*  NOPAL TEST SPECIFICATION FOR CPS#10559261      */
   /*                                                */
   /**************************************************/

NOPAL SPECIFICATION CPS#10559261;

   /**************************************************/
   /*                                                */
   /*  TEST MODULES:      7                           */
   /*                                                */
   /**************************************************/


 TEST SYTEST0001;

    /* NULL STIMULI */

    /* NULL MEASUREMENT */

    LOGIC $LOGIC0010(SYTEST0001): *1;

      DIAGNOSIS 1:
          OPERATOR MESSAGE:
              TYPE=X;

 TEST 55111;

    STIMULI S55111(55111);

       CONJUNCTION $S_W0001(S55111):
           ((<J1_G, J1_B> VOLT = P_SUPPLY(29 VOLT ,1000 MAMP )) &
           ((<J1_F, J1_B> VOLT = P_SUPPLY(15 VOLT ,1 MAMP )) &
           ((<J1_H, J1_B> VOLT = P_SUPPLY(24,100 MAMP )) &
           ((<J1_A, J1_B>  = LOAD_L(2200 OHM ));
```

TABLE B.3 NOPAL FORMATTED LISTING OF TEST MODULES SPECIFICATION

```
MEASUREMENT $M_55111(55111);

   CONJUNCTION $M_W0001($M_55111):
       (<J1_G> MAMP = AMP_M(IE29V MAMP ))
              TARGET: IE29V;

   ASSERTION $M_W0002($M-55111):
       ABS(IE29V-47) <= 10
              SOURCE: IE29V;

LOGIC $LOGIC0010(55111): |¬2, &¬8, &¬10, &¬12;

  DIAGNOSIS 2:
     OPERATOR MESSAGE:
        AFFECTED COMPONENTS=S(R1)| S(R11)| S(CR3)| NS(Q1B),
        TYPE=1;

  DIAGNOSIS 8:
     OPERATOR MESSAGE:
        AFFECTED COMPONENTS=O(R9)| O(R11)| S(R9)| S(R10)| ON(Q4)|
            OO(G4)| OS(Q4)| NS(Q4)| SN(Q4)| SO(Q4),
        TYPE=1;

 DIAGNOSIS 10:
    OPERATOR MESSAGE:
       AFFECTED COMPONENTS=O(R1)| O(R2)| O(R5)| O(CR2)| S(R3)| S(R4)|
           ON(Q1A)| OO(Q1A)| OS(Q1A)| NS(Q1A)| NO(Q1A)| SS(Q1B)|
           SS(Q2)| SN(Q1B)| SN(Q2),
       TYPE=1;

 DIAGNOSIS 12:
    OPERATOR MESSAGE:
       AFFECTED COMPONENTS=SO(Q1A),
       TYPE=1;
```

TABLE B.3 NOPAL FORMATTED LISTING OF TEST MODULES SPECIFICATION (continued)

```
TEST 55112;

  STIMULI $S_55112(55112);

    CONJUNCTION $S_W0001($S_55112):
        (<J1_G, J1_B> VOLT = P_SUPPLY(29 VOLT ,1000 MAMP )) &
        (<J1_F, J1_B> VOLT = P_SUPPLY(15 VOLT ,1 MAMP )) &
        (<J1_H, J1_B> VOLT = P_SUPPLY(24,100 MAMP )) &
        (<J1_A, J1_B>  = LOAD_L(220n OHM )) &
        (<S1>  = COOL(14 DEGF ));

  MEASUREMENT $M_55112(55112);

    CONJUNCTION $M_W0001($M_55112):
        (<J1_3>  = AMP_M(IE29VL MAMP ))
                TARGET: IE29VL;

    ASSERTION $M_W0002($M_55112):
        IE29V-IE29VL >  10
                SOURCE: IE29VL, IE29V;

  LOGIC $LOGIC0010(55112): |¬3, &¬8;

   DIAGNOSIS 3:
     OPERATOR MESSAGE:
       AFFECTED COMPONENTS=O(CR3)| SS(Q4)| NO(Q4),
       TYPE=1;

  /*** FOLLOWING DIAGNOSIS ALREADY DEFINED BEFORE:

   DIAGNOSIS 8:
     OPERATOR MESSAGE:
       AFFECTED COMPONENTS=O(R9)| O(R11)| S(R9)| S(R10)| ON(Q4)|
           OO(Q4)| OS(Q4)| NS(Q4)| SN(Q4)| SO(Q4),
       TYPE=1;
                                                     ***/
```

TABLE B.3 NOPAL FORMATTED LISTING OF TEST MODULES SPECIFICATION (continued)

```
TEST 55121;

  STIMULI $S_55121(55121);

    CONJUNCTION $S_W0001($S_55121):
        (<J1_G, J1_B> VOLT = P_SUPPLY(29 VOLT ,1000 MAMP )) &
        (<J1_F, J1_B> VOLT = P_SUPPLY(15 VOLT ,1 MAMP )) &
        (<J1_H, J1_B> VOLT = P_SUPPLY(31 VOLT ,10 MAMP )) &
        (<J1_A, J1_B>  = LOAD_L(220n OHM ));

  MEASUREMENT $M_55121(55121);

    CONJUNCTION $M_W0001($M_55121):

        (<J1_A, J1_B>  = VOLT_M(VNJ1A VOLT ))
               TARGET: VNJ1A;

    ASSERTION $M_W0002($M_55121):
        ABS(VNJ1A-30.5) <= 1.5
               SOURCE: VNJ1A;

  LOGIC $LOGIC0010(55121): I¬4;

    DIAGNOSIS 4:
        OPERATOR MESSAGE:
            AFFECTED COMPONENTS=o(CR4),
            TYPE=1;

TEST 55122;

  STIMULI $S_55122(55122);

    CONJUNCTION $S_W0001($S_55122):
        (<J1_G, J1_B> VOLT = P_SUPPLY(29 VOLT ,1000 MAMP )) &
        (<J1_F, J1_B> VOLT = P_SUPPLY(15 VOLT ,1 MAMP )) &
        (<J1_H, J1_B> VOLT = P_SUPPLY(15 VOLT ,10 MAMP )) &
        (<J1_A, J1_B>  = LOAD_L(220n OHM ));
```

TABLE B.3 NOPAL FORMATTED LISTING OF TEST MODULES SPECIFICATION (continued)

```
MEASUREMENT $M_55122(55122);

   CONJUNCTION $M_W0001($M_55122):
      (<J1_A, J1_B> VOLT = VOLT_M(VNJ1A VOLT ))
            TARGET: VNJ1A;

   ASSERTION $M_W0002($M_55122):
      ABS(VNJ1A-23) <  6
            SOURCE: VNJ1A;

LOGIC $LOGIC0010(55122): !¬5;

  DIAGNOSIS 5:
    OPERATOR MESSAGE:
      AFFECTED COMPONENTS=0(CR5),
      TYPE=1;

TEST 55131;

STIMULI S55131(55131);

   CONJUNCTION $S_W0001(S55131):
      (<J1_G, J1_B> VOLT = P_SUPPLY(29 VOLT ,1000 MAMP )) &
      (<J1_F, J1_B> VOLT = P_SUPPLY(15 VOLT ,1 MAMP )) &
      (<J1_H, J1_B> VOLT = P_SUPPLY(24 VOLT ,10 MAMP )) &
      (<J1_A, J1_B>  = LOAD_L(2200 OHM )) &
      (<J1_E, J1_B>  = LOAD_NL);

MEASUREMENT $M_55131(55131);

   CONJUNCTION $M_W0001($M_55131):
      (<J1_240, J1_B> VOLT = VOLT_M(VR240 VOLT ))
            TARGET: VR240;

   ASSERTION $M_W0002($M-55131):
      ABS(VR240-16) <= 2
            SOURCE: VR240;

LOGIC $LOGIC0010(55131): !¬6, &¬9, &¬10, &¬12;
```

TABLE B.3 NOPAL FORMATTED LISTING OF TEST MODULES SPECIFICATION (continued)

```
DIAGNOSIS 6:
    OPERATOR MESSAGE:
        AFFECTED COMPONENTS=SS(Q3)| SN(Q3),
        TYPE=1;

DIAGNOSIS 9:
    OPERATOR MESSAGE:
        AFFECTED COMPONENTS=O(R3),
        TYPE=1;

/*** FOLLOWING DIAGNOSIS ALREADY DEFINED BEFORE:

DIAGNOSIS 10:
    OPERATOR MESSAGE:
        AFFECTED COMPONENTS=O(R1)| O(R2)| O(R5)| O(CR2)| S(R3)| S(R4)|
            ON(Q1A)| OO(Q1A)| OS(Q1A)| NS(Q1A)| NO(Q1A)| SS(Q1B)|
            SS(Q2)| SN(Q1B)| SN(Q2),
        TYPE=1;
                                                              ***/

/*** FOLLOWING DIAGNOSIS ALREADY DEFINED BEFORE:

DIAGNOSIS 12:
    OPERATOR MESSAGE:
        AFFECTED COMPONENTS=SO(Q1A),
        TYPE=1;
                                                              ***/

TEST 55132:

  STIMULI $S_55132(55132):

    CONJUNCTION $S_W0001($S_55132):
        (<J1_G, J1_B> VOLT = P_SUPPLY(15 VOLT ,10 MAMP )) &
        (<J1_F, J1_B> VOLT = P_SUPPLY(15 VOLT ,1 MAMP )) &
        (<J1_H, J1_B> VOLT = P_SUPPLY(24 VOLT ,10 MAMP )) &
        (<J1_A, J1_B>  = LOAD_L(220n OHM )) &
        (<J1_E, J1_B>  = LOAD_NL);
```

TABLE B.3 NOPAL FORMATTED LISTING OF TEST MODULES SPECIFICATION (continued)

```
MEASUREMENT $M_55132(55132);

   CONJUNCTION $M_W0001($M_55132):
       (<J1_A, J1_B>  = VOLT_M(VR240 VOLT )) &
       (<J1_2200, J1_B> VOLT = VOLT_M(VR2200 VOLT ))
              TARGET: VR2200, VR240;

   ASSERTION $M_W0002($M_55132):
       ABS(VR240-VR2200) <= 2
              SOURCE: VR240, VR2200;

LOGIC $LOGIC0010(55132): |¬7, &¬9, &¬11, &¬12;

  DIAGNOSIS 7:
      OPERATOR MESSAGE:
          AFFECTED COMPONENTS=O(R3)| O(R8)| O(CR1)| S(C1)| SS(Q1A)|
              SN(Q1A)| NO(Q1B)| NO(Q2)| NO(Q3)| OO(Q2)| OO(Q3)| SO(Q3),
          TYPE=1;

 /*** FOLLOWING DIAGNOSIS ALREADY DEFINED BEFORE:

  DIAGNOSIS 9:
      OPERATOR MESSAGE:



         AFFECTED COMPONENTS=O(R3),
         TYPE=1;
                                                          ***/

 DIAGNOSIS 11:
     OPERATOR MESSAGE:
         AFFECTED COMPONENTS=S(R2)| S(R5)| SO(Q2),
         TYPE=1;

/*** FOLLOWING DIAGNOSIS ALREADY DEFINED BEFORE:

 DIAGNOSIS 12:
     OPERATOR MESSAGE:
         AFFECTED COMPONENTS=SO(Q1A),
         TYPE=1;
                                                          ***/
```

TABLE B.3 NOPAL FORMATTED LISTING OF TEST MODULES SPECIFICATION (continued)

```
/*****************************************************/
/*                                                   */
/*  MESSAGES                                         */
/*                                                   */
/*****************************************************/

MESSAGE X:
   TEXT='SET UP AS REQUIRED IN CPS#10559261.';

MESSAGE 1:
   TEXT='FAULTY COMPONENTS: (C)';
```

TABLE B.3 NOPAL FORMATTED LISTING OF TEST MODULES SPECIFICATION (continued)

```
/**************************************************/
/*                                                */
/*   UUT COMPONENTS/FAILURES                      */
/*                                                */
/**************************************************/

COMP_FAIL 1: R1, FAILURE FUNCTION=O;

COMP_FAIL 2: R1, FAILURE FUNCTION=S;

COMP_FAIL 3: R2, FAILURE FUNCTION=O;

COMP_FAIL 4: R2, FAILURE FUNCTION=S;

COMP_FAIL 5: R3, FAILURE FUNCTION=O;

COMP_FAIL 6: R3, FAILURE FUNCTION=S;

COMP_FAIL 7: R4, FAILURE FUNCTION=O;

COMP_FAIL 8: R4, FAILURE FUNCTION=S;

COMP_FAIL 9: R5, FAILURE FUNCTION=O;

COMP_FAIL 10: R5, FAILURE FUNCTION=S;

COMP_FAIL 11: R6, FAILURE FUNCTION=O;

COMP_FAIL 12: R6, FAILURE FUNCTION=S;

COMP_FAIL 13: R7, FAILURE FUNCTION=O;
COMP_FAIL 14: R7, FAILURE FUNCTION=S;

COMP_FAIL 16: R8, FAILURE FUNCTION=S;

COMP_FAIL 15: R8, FAILURE FUNCTION=O;

COMP_FAIL 17: R9, FAILURE FUNCTION=O;
```

TABLE B.4 NOPAL FORMATTED LISTING OF UUT SPECIFICATION

```
COMP_FAIL 18: R9, FAILURE FUNCTION=S;
COMP_FAIL 20: R10, FAILURE FUNCTION=S;
COMP_FAIL 19: R10, FAILURE FUNCTION=O;
COMP_FAIL 21: R11, FAILURE FUNCTION=O;
COMP_FAIL 22: R11, FAILURE FUNCTION=S;
COMP_FAIL 23: C1, FAILURE FUNCTION=O;
COMP_FAIL 24: C1, FAILURE FUNCTION=S;
COMP_FAIL 25: CR1, FAILURE FUNCTION=O;
COMP_FAIL 26: CR1, FAILURE FUNCTION=S;
COMP_FAIL 28: CR2, FAILURE FUNCTION=S;
COMP_FAIL 27: CR2, FAILURE FUNCTION=O;
COMP_FAIL 29: CR3, FAILURE FUNCTION=O;
COMP_FAIL 30: CR3, FAILURE FUNCTION=S;
COMP_FAIL 32: CR4, FAILURE FUNCTION=S;
COMP_FAIL 31: CR4, FAILURE FUNCTION=O;
COMP_FAIL 33: CR5, FAILURE FUNCTION=O;
COMP_FAIL 34: CR5, FAILURE FUNCTION=S;
COMP_FAIL 03500: Q1A, FAILURE FUNCTION=ON;
COMP_FAIL 03600: Q1A, FAILURE FUNCTION=OO;
COMP_FAIL 03700: Q1A, FAILURE FUNCTION=OS;
```

TABLE B.4 NOPAL FORMATTED LISTING OF UUT SPECIFICATION (continued)

```
COMP_FAIL 03800: Q1A, FAILURE FUNCTION=NS;
COMP_FAIL 03900: Q1A, FAILURE FUNCTION=SS;
COMP_FAIL 04000: Q1A, FAILURE FUNCTION=SN;
COMP_FAIL 04100: Q1A, FAILURE FUNCTION=SO;
COMP_FAIL 04200: Q1A, FAILURE FUNCTION=NO;
COMP_FAIL 04300: Q1B, FAILURE FUNCTION=ON;
COMP_FAIL 04400: Q1B, FAILURE FUNCTION=OO;
COMP_FAIL 04500: Q1B, FAILURE FUNCTION=OS;
COMP_FAIL 04600: Q1B, FAILURE FUNCTION=NS;
COMP_FAIL 04700: Q1B, FAILURE FUNCTION=SS;
COMP_FAIL 04800: Q1B, FAILURE FUNCTION=SN;
COMP_FAIL 04900: Q1B, FAILURE FUNCTION=SO;
COMP_FAIL 05000: Q1B, FAILURE FUNCTION=NO;
COMP_FAIL 05100: Q2, FAILURE FUNCTION=ON;
COMP_FAIL 05200: Q2, FAILURE FUNCTION=OO;
COMP_FAIL 05300: Q2, FAILURE FUNCTION=OS;
COMP_FAIL 05400: Q2, FAILURE FUNCTION=NS;
COMP_FAIL 05500: Q2, FAILURE FUNCTION=SS;
COMP_FAIL 05600: Q2, FAILURE FUNCTION=SN;
COMP_FAIL 05700: Q2, FAILURE FUNCTION=SO;
```

TABLE B.4 NOPAL FORMATTED LISTING OF UUT SPECIFICATION (continued)

```
COMP_FAIL 05800: Q2, FAILURE FUNCTION=NO;
COMP_FAIL 05900: Q3, FAILURE FUNCTION=ON;
COMP_FAIL 06000: Q3, FAILURE FUNCTION=OO;
COMP_FAIL 06100: Q3, FAILURE FUNCTION=OS;
COMP_FAIL 06200: Q3, FAILURE FUNCTION=NS;
COMP_FAIL 06300: Q3, FAILURE FUNCTION=SS;
COMP_FAIL 06400: Q3, FAILURE FUNCTION=SN;
COMP_FAIL 06500: Q3, FAILURE FUNCTION=SO;
COMP_FAIL 06600: Q3, FAILURE FUNCTION=NO;
COMP_FAIL 06700: Q4, FAILURE FUNCTION=ON;
COMP_FAIL 06800: Q4, FAILURE FUNCTION=OO;
COMP_FAIL 06900: Q4, FAILURE FUNCTION=OS;
COMP_FAIL 07000: Q4, FAILURE FUNCTION=NS;
COMP_FAIL 07100: Q4, FAILURE FUNCTION=SS;
COMP_FAIL 07200: Q4, FAILURE FUNCTION=SN;
COMP_FAIL 07300: Q4, FAILURE FUNCTION=SO;
COMP_FAIL 07400: Q4, FAILURE FUNCTION=NO;
```

TABLE B.4 NOPAL FORMATTED LISTING OF UUT SPECIFICATION (continued)

```
/****************************************************/
/*                                                  */
/*   UUT CONNECTION POINTS                          */
/*                                                  */
/****************************************************/

UUT_POINT      : J1_G, ALIAS=E29V, LIMIT=(VOLT,  1.50000E+01,  0.00000E+00
    , J1_B), COMMENTS='29V MAIN POWER';

UUT_POINT      : J1_B, ALIAS=TANK_POWER_B, COMMENTS='GROUND';

UUT_POINT     1: J1_F, ALIAS=E15V, LIMIT=(VOLT,  1.50000E+01,  0.00000E+00
    , J1_B), COMMENTS='15V BIASING REF.';

UUT_POINT      : J1_H, ALIAS=TANK_BATTERY,
        LIMIT=(VOLT,  3.10000E+01,  0.00000E+00, J1_B),
        COMMENTS='SENSE SIGNAL FROM MAIN POWER SUPPLY.';

UUT_POINT      : J1_A, ALIAS=G_SWITCH,
        LIMIT=(VOLT,  3.10000E+01,  0.00000E+00, J1_B),
        COMMENTS='24V WHEN SENSE<24; ELSE SENSE.';

UUT_POINT      : S1, CONNECTOR=(AIR_OR_CONTA, ),
        LIMIT=(DEGF,  7.50000E+01,  1.40000E+01),
        COMMENTS='COLD AIR ON EQ.';

UUT_POINT      : J1_E, ALIAS=AUX_POWER,
        LIMIT=(VOLT,  3.10000E+01,  0.00000E+00, J1_B),
        COMMENTS='IF VNJ1G=29V & VNJ1H=30V THEN 15V; ELSE 30V.';

UUT_POINT      : J1_240, COMMENTS='DUMMY POINT';

UUT_POINT      : J1_2200, ALIAS=LOADRESISTOR;

UUT_POINT      : J1_C, ALIAS=EXT_BATTERY_,
        LIMIT=(VOLT,  2.40000E+01,  0.00000E+00, J1_B),
        COMMENTS='USE TO REPLACE BT1 OR TO CHARGE BT1';
```

TABLE B.4 NOPAL FORMATTED LISTING OF UUT SPECIFICATION (continued)

```
/************************************************/
/*                                              */
/*   ATE FUNCTIONS                              */
/*                                              */
/************************************************/

FUNCTION       : P_SUPPLY, ALIAS=DC_POWER_SUP, FUNCTION TYPE=S, #PINS= 2,
       PARAM_01=(V, S, LIMIT=(VOLT,  3.40000E+01,  0.00000E+00)),
       PARAM_02=(I, S, LIMIT=(AMP,  2.00000E+00,  0.00000E+00));

FUNCTION       : LOAD_L, ALIAS=LOAD_LINEAR, FUNCTION TYPE=S, #PINS= 2,
       PARAM_01=(R, S, LIMIT=(KOHM,  1.00000E+01,  0.00000E+00));

FUNCTION       : AMP_M, ALIAS=AMPEREMETER, FUNCTION TYPE=M, #PINS= 1,
       PARAM_01=(I, T, LIMIT=(AMP,  5.00000E+00, -5.00000E+00)),
       VALUE RETURNED='CNST. CURRENT', COMMENTS='+ READING I INTO UUT';

FUNCTION       : COOL, ALIAS=COOLANT, FUNCTION TYPE=S, #PINS= 1,
       PARAM_01=(DESIR, S, LIMIT=(DEGF,  7.60000E+01,  1.50000E+01)),
       COMMENTS='COOL UNIT UNTIL DESIRED TEMPERATURE.';

FUNCTION     10: VOLT_M, ALIAS=VOLTMETER, FUNCTION TYPE=M, #PINS= 2,
       PARAM_01=(V, T, LIMIT=(VOLT,  3.10000E+01, -3.10000E+01)),
       VALUE RETURNED='CONST VOLT.', COMMENTS='V-READING';

FUNCTION       : LOAD_NL, ALIAS=NONLINEAR, FUNCTION TYPE=S, #PINS= 2;

FUNCTION       : S, FUNCTION TYPE=F,
       PARAM_01=(COMP, S);

FUNCTION       : VS, FUNCTION TYPE=F,
       PARAM_01=(COMP, S);

FUNCTION       : D, FUNCTION TYPE=F,
       PARAM_01=(COMPONENT, S);

FUNCTION       : DV, FUNCTION TYPE=F,
       PARAM_01=(COMP, S);
```

TABLE B.5 NOPAL FORMATTED LISTING OF ATE SPECIFICATION

```
FUNCTION        : DD, FUNCTION TYPE=F,
        PARAM_01=(COMP, S);

FUNCTION        : DS, FUNCTION TYPE=F,
        PARAM_01=(COMP, S);

FUNCTION        : SV, FUNCTION TYPE=F,
        PARAM_01=(COMP, S);

FUNCTION        : SD, FUNCTION TYPE=F,
        PARAM_01=(COMP, S);

FUNCTION        : VD, FUNCTION TYPE=F,
        PARAM_01=(COMP, S);

FUNCTION        : SS, FUNCTION TYPE=F,
        PARAM_01=(COMP, S);

FUNCTION        : ABS, ALIAS=ABSOLUTE, FUNCTION TYPE=E,
        PARAM_01=(EXP, S), COMMENTS='TAKE ABS. VALUE OF EXP';

END CPS#10559261;
```

TABLE B.5 NOPAL FORMATTED LISTING OF ATE SPECIFICATION (continued)

```
ERROR/WARNING MESSAGES GENERATED DURING NOPAL SYNTAX ANALYSIS:

*WARNING*  IN STATEMENT  134, NEAR TEXT 'EXT_BATTERY_', NAME/INTEGER WAS TOO LONG.  TRUNCATED.
*WARNING*  IN STATEMENT  135, NEAR TEXT 'TANK_BATTERY', NAME/INTEGER WAS TOO LONG.  TRUNCATED.
*WARNING*  IN STATEMENT  136, NEAR TEXT 'TANK_POWER_R', NAME/INTEGER WAS TOO LONG.  TRUNCATED.
*WARNING*  IN STATEMENT  139, NEAR TEXT 'AIR_OR_CONTA', NAME/INTEGER WAS TOO LONG.  TRUNCATED.
*WARNING*  IN STATEMENT  145, NEAR TEXT 'DC_POWER_SUP', NAME/INTEGER WAS TOO LONG.  TRUNCATED.

*STATISTICS*  NO. OF SAP ERRORS =    0,    NO. OF WARNINGS =    5,    NO. OF STATEMENTS =  159
```

TABLE B.6 ERROR/WARNING DURING NOPAL SYNTAX ANALYSIS

```
NAME           DEF NO.     ATTRIBUTES AND REFERENCES
----           -------     -------------------------
ABS              148       ATE-FUNCTION ID, E
                             10  23  30  36  43
ABSOLUTE         148       SYNONYM OF ATE-FUNCTION ID: ABS, E
AMP_M            143       ATE-FUNCTION ID, M
                             9  16
AMPEREMETER      143       SYNONYM OF ATE-FUNCTION ID: AMP_M, M
AUX_POWER        137       SYNONYM OF UUT-POINT ID: J1-E
COOL             144       ATE-FUNCTION ID, S
                            14
COOLANT          144       SYNONYM OF ATE-FUNCTION ID: COOL, S
CPS#10559261       1       SPECIFICATION LABEL
                            159
CR1               73       COMPONENT ID, WITH FAILURE-FUNCTION: O
CR1               74       COMPONENT ID, WITH FAILURE-FUNCTION: S
CR2               75       COMPONENT ID, WITH FAILURE-FUNCTION: S
CR2               76       COMPONENT ID, WITH FAILURE-FUNCTION: O
                             47
CR3               77       COMPONENT ID, WITH FAILURE-FUNCTION: O
                            124
CR3               78       COMPONENT ID, WITH FAILURE-FUNCTION: S
                             45
CR4               79       COMPONENT ID, WITH FAILURE-FUNCTION: S
CR4               80       COMPONENT ID, WITH FAILURE-FUNCTION: O
                            125
CR5               81       COMPONENT ID, WITH FAILURE-FUNCTION: O
                            126
CR5               82       COMPONENT ID, WITH FAILURE-FUNCTION: S
C1                71       COMPONENT ID, WITH FAILURE-FUNCTION: O
C1                72       COMPONENT ID, WITH FAILURE-FUNCTION: S
DC_POWER_SUP     145       SYNONYM OF ATE-FUNCTION ID: P_SUPPLY, S
EXT_BATTERY_     134       SYNONYM OF UUT-POINT ID: J1-C
E15V             132       SYNONYM OF UUT-POINT ID: J1-F
E29V             133       SYNONYM OF UUT-POINT ID: J1-G
IE29V              9       VARIABLE ID, GLOBAL
                             10  17
IE29VL            16       VARIABLE ID
                             17
J1-A             138       UUT-POINT ID
                             7  22  29  42  14  20  27  33  40
J1-B             136       UUT-POINT ID
                             7  22  29  35  42 132 133 134 135 137 138  14  20  27  33  40
J1-C             134       UUT-POINT ID
J1-E             137       UUT-POINT ID
                            33  40
J1-F             132       UUT-POINT ID
                             7  14  20  27  33  40
J1-G             133       UUT-POINT ID
                             7   9  16  14  20  27  33  40
J1-H             135       UUT-POINT ID
                             7  14  20  27  33  40
J1-2200          140       UUT-POINT ID
                            42
J1-240           141       UUT-POINT ID
                            35
LOAD_L           146       ATE-FUNCTION ID, S
                             7  14  20  27  33  40
```

TABLE B.7 CROSS REFERENCE AND ATTRIBUTE REPORT

```
LOAD_LINEAR     146     SYNONYM OF ATE-FUNCTION ID: LOAD_L, S
LOAD_NL         147     ATE-FUNCTION ID, S
                          33  40
LOADRESISTOR    140     SYNONYM OF UUT-POINT ID: J1-2200
NO              158     ATE-FUNCTION ID, F
                          47  90  98 106 114 122 124 129
NONLINEAR       147     SYNONYM OF ATE-FUNCTION ID: LOAD_NL, S
NS              154     ATE-FUNCTION ID, F
                          45  46  47  86  94 102 110 118
O               149     ATE-FUNCTION ID, F
                          46  47  49  51  53  55  57  59  61  64  65  68  69  71  73  76  77  80  81 124 125 126 128
ON              151     ATE-FUNCTION ID, F
                          46  47  83  91  99 107 115
OO              152     ATE-FUNCTION ID, F
                          46  47  84  92 100 108 116 129
OS              153     ATE-FUNCTION ID, F
                          46  47  85  93 101 109 117
P_SUPPLY        145     ATE-FUNCTION ID, S
                           7  14  20  27  33  40
Q_SWITCH        138     SYNONYM OF UUT-POINT ID: J1-A
Q1A              83     COMPONENT ID, WITH FAILURE-FUNCTION: ON
                          47
Q1A              84     COMPONENT ID, WITH FAILURE-FUNCTION: OO
                          47
Q1A              85     COMPONENT ID, WITH FAILURE-FUNCTION: OS
                          47
Q1A              86     COMPONENT ID, WITH FAILURE-FUNCTION: NS
                          47
Q1A              87     COMPONENT ID, WITH FAILURE-FUNCTION: SS
                         129
Q1A              88     COMPONENT ID, WITH FAILURE-FUNCTION: SN
                         129
Q1A              89     COMPONENT ID, WITH FAILURE-FUNCTION: SO
                          48
Q1A              90     COMPONENT ID, WITH FAILURE-FUNCTION: NO
                          47
Q1B              91     COMPONENT ID, WITH FAILURE-FUNCTION: ON
Q1B              92     COMPONENT ID, WITH FAILURE-FUNCTION: OO
Q1B              93     COMPONENT ID, WITH FAILURE-FUNCTION: OS
Q1B              94     COMPONENT ID, WITH FAILURE-FUNCTION: NS
                          45
Q1B              95     COMPONENT ID, WITH FAILURE-FUNCTION: SS
                          47
Q1B              96     COMPONENT ID, WITH FAILURE-FUNCTION: SN
                          47
Q1B              97     COMPONENT ID, WITH FAILURE-FUNCTION: SO
Q1B              98     COMPONENT ID, WITH FAILURE-FUNCTION: NO
                         129
Q2               99     COMPONENT ID, WITH FAILURE-FUNCTION: ON
Q2              100     COMPONENT ID, WITH FAILURE-FUNCTION: OO
                         129
Q2              101     COMPONENT ID, WITH FAILURE-FUNCTION: OS
Q2              102     COMPONENT ID, WITH FAILURE-FUNCTION: NS
Q2              103     COMPONENT ID, WITH FAILURE-FUNCTION: SS
                          47
Q2              104     COMPONENT ID, WITH FAILURE-FUNCTION: SN
                          47
Q2              105     COMPONENT ID, WITH FAILURE-FUNCTION: SO
                         123
Q2              106     COMPONENT ID, WITH FAILURE-FUNCTION: NO
```

TABLE B.7 CROSS REFERENCE AND ATTRIBUTE REPORT (continued)

```
                                  129
Q3                 107            COMPONENT ID, WITH FAILURE-FUNCTION: ON
Q3                 108            COMPONENT ID, WITH FAILURE-FUNCTION: OD
                                  129
Q3                 109            COMPONENT ID, WITH FAILURE-FUNCTION: OS
Q3                 110            COMPONENT ID, WITH FAILURE-FUNCTION: NS
Q3                 111            COMPONENT ID, WITH FAILURE-FUNCTION: SS
                                  127
Q5                 112            COMPONENT ID, WITH FAILURE-FUNCTION: SN
                                  127
Q3                 113            COMPONENT ID, WITH FAILURE-FUNCTION: SO
                                  129
Q3                 114            COMPONENT ID, WITH FAILURE-FUNCTION: NO
                                  129
Q4                 115            COMPONENT ID, WITH FAILURE-FUNCTION: ON
                                  46
Q4                 116            COMPONENT ID, WITH FAILURE-FUNCTION: OO
                                  46
Q4                 117            COMPONENT ID, WITH FAILURE-FUNCTION: OS
                                  46
Q4                 118            COMPONENT ID, WITH FAILURE-FUNCTION: NS
                                  46
Q4                 119            COMPONENT ID, WITH FAILURE-FUNCTION: SS
                                  124
Q4                 120            COMPONENT ID, WITH FAILURE-FUNCTION: SN
                                  46
Q4                 121            COMPONENT ID, WITH FAILURE-FUNCTION: SO
                                  46
Q4                 122            COMPONENT ID, WITH FAILURE-FUNCTION: NO
                                  124
R1                  49            COMPONENT ID, WITH FAILURE-FUNCTION: O
                                  47
R1                  50            COMPONENT ID, WITH FAILURE-FUNCTION: S
                                  45
R10                 67            COMPONENT ID, WITH FAILURE-FUNCTION: S
                                  46
R10                 68            COMPONENT ID, WITH FAILURE-FUNCTION: O
R11                 69            COMPONENT ID, WITH FAILURE-FUNCTION: O
                                  46
R11                 70            COMPONENT ID, WITH FAILURE-FUNCTION: S
                                  45
R2                  51            COMPONENT ID, WITH FAILURE-FUNCTION: O
                                  47
R2                  52            COMPONENT ID, WITH FAILURE-FUNCTION: S
R3                  53            COMPONENT ID, WITH FAILURE-FUNCTION: O
                                  128
R3                  54            COMPONENT ID, WITH FAILURE-FUNCTION: S
                                  47
R4                  55            COMPONENT ID, WITH FAILURE-FUNCTION: O
R4                  56            COMPONENT ID, WITH FAILURE-FUNCTION: S
                                  47
R5                  57            COMPONENT ID, WITH FAILURE-FUNCTION: O
                                  47
R5                  58            COMPONENT ID, WITH FAILURE-FUNCTION: S
R6                  59            COMPONENT ID, WITH FAILURE-FUNCTION: O
R6                  60            COMPONENT ID, WITH FAILURE-FUNCTION: S
R7                  61            COMPONENT ID, WITH FAILURE-FUNCTION: O
R7                  62            COMPONENT ID, WITH FAILURE-FUNCTION: S
R8                  63            COMPONENT ID, WITH FAILURE-FUNCTION: S
R8                  64            COMPONENT ID, WITH FAILURE-FUNCTION: O
```

TABLE B.7 CROSS REFERENCE AND ATTRIBUTE REPORT (continued)

```
R9                 65      COMPONENT ID, WITH FAILURE-FUNCTION: O
                             46
R9                 66      COMPONENT ID, WITH FAILURE-FUNCTION: S
                             46
S                 150      ATE-FUNCTION ID, F
                             45  46  47  50  52  54  56  58  60  62  63  66  67  70  72  74  75  78  79  82
SN                156      ATE-FUNCTION ID, F
                             46  47  88  96 104 112 120 127 129
SO                157      ATE-FUNCTION ID, F
                             46  48  89  97 105 113 121 123 129
SS                155      ATE-FUNCTION ID, F
                             47  87  95 103 111 119 124 127 129
S1                139      UUT-POINT ID
                             14
S55111              6      STIMULUS LABEL
                              7  14  20  27  33
S55131             33      STIMULUS LABEL
                             33  40
TANK_BATTERY      135      SYNONYM OF UUT-POINT ID: J1-H
TANK POWER_R      136      SYNONYM OF UUT-POINT ID: J1-B
VNJ1A              22      VARIABLE ID
                             23
VNJ1A              29      VARIABLE ID
                             30
VOLT_M            142      ATE-FUNCTION ID, M
                             22  29  35  42
VOLTMETER         142      SYNONYM OF ATE-FUNCTION ID: VOLT_M, M
VR2200             42      VARIABLE ID
                             43
VR240              35      VARIABLE ID
                             36
VR240              42      VARIABLE ID
                             43
X                 130      MESSAGE LABEL
                             4
1                  49      COMPONENT/FAILURE SEQ#
1                 131      MESSAGE LABEL
                             45  46  47  48 123 124 125 126 127 128 129
1                   4      DIAGNOSIS LABEL
                             3
10                 58      COMPONENT/FAILURE SEQ#
                             123
10                 47      DIAGNOSIS LABEL
                             11  37
11                 59      COMPONENT/FAILURE SEQ#
11                123      DIAGNOSIS LABEL
                             44
12                 60      COMPONENT/FAILURE SEQ#
12                 48      DIAGNOSIS LABEL
                             11  37  44
13                 61      COMPONENT/FAILURE SEQ#
14                 62      COMPONENT/FAILURE SEQ#
15                 64      COMPONENT/FAILURE SEQ#
                             129
16                 63      COMPONENT/FAILURE SEQ#
17                 65      COMPONENT/FAILURE SEQ#
18                 66      COMPONENT/FAILURE SEQ#
19                 68      COMPONENT/FAILURE SEQ#
2                  50      COMPONENT/FAILURE SEQ#
2                  45      DIAGNOSIS LABEL
```

TABLE B.7 CROSS REFERENCE AND ATTRIBUTE REPORT (continued)

		11
20	67	COMPONENT/FAILURE SEQ#
21	69	COMPONENT/FAILURE SEQ#
22	70	COMPONENT/FAILURE SEQ#
23	71	COMPONENT/FAILURE SEQ#
24	72	COMPONENT/FAILURE SEQ#
		129
25	73	COMPONENT/FAILURE SEQ#
		129
26	74	COMPONENT/FAILURE SEQ#
27	76	COMPONENT/FAILURE SEQ#
28	75	COMPONENT/FAILURE SEQ#
29	77	COMPONENT/FAILURE SEQ#
3	51	COMPONENT/FAILURE SEQ#
3	124	DIAGNOSIS LABEL
		18
30	78	COMPONENT/FAILURE SEQ#
31	80	COMPONENT/FAILURE SEQ#
32	79	COMPONENT/FAILURE SEQ#
33	81	COMPONENT/FAILURE SEQ#
34	82	COMPONENT/FAILURE SEQ#
4	52	COMPONENT/FAILURE SEQ#
		123
4	125	DIAGNOSIS LABEL
		24
5	53	COMPONENT/FAILURE SEQ#
		129
5	126	DIAGNOSIS LABEL
		31
55111	5	TEST LABEL
		6 8 11
55112	12	TEST LABEL
		13 15 18
55121	19	TEST LABEL
		20 21 24
55122	25	TEST LABEL
		26 28 31
55131	32	TEST LABEL
		33 34 37
55132	38	TEST LABEL
		39 41 44
6	54	COMPONENT/FAILURE SEQ#
6	127	DIAGNOSIS LABEL
		37
7	55	COMPONENT/FAILURE SEQ#
7	129	DIAGNOSIS LABEL
		44
8	56	COMPONENT/FAILURE SEQ#
8	46	DIAGNOSIS LABEL
		11 18
9	57	COMPONENT/FAILURE SEQ#
9	128	DIAGNOSIS LABEL
		37 44

TABLE B.7 CROSS REFERENCE AND ATTRIBUTE REPORT (continued)

```
ERROR/WARNING MESSAGES GENERATED DURING CROSS-REFERENCE:

*STATISTICS*  NO. OF XREF1 ERRORS =     0     NO. OF WARNINGS =     0
```

TABLE B.8 ERROR/WARNING DURING CROSS-REFERENCE

SUMMARY CROSS-REFERENCES, FILE: XREF2 --- DIAGNOSES <=> TEST-MODULES

DIAGNOSIS	TEST-MODULES
1	SYTEST0001
2	55111
8	55111, 55112
10	55111, 55131
12	55111, 55131, 55132
3	55112
4	55121
5	55122
6	55131
9	55131, 55132
7	55132
11	55132

TABLE B.9 CROSS-REFERENCE --- DIAGNOSES TO TESTS

SUMMARY CROSS-REFERENCES, FILE: XREF2 --- MESSAGES <=> DIAGNOSES <=> TESTS

MESSAGE	DIAGNOSES	TEST-MODULES
X	1	SYTEST0001
1	2, 8, 1n, 12, 11, 3, 4, 5, 6, 9, 7	55111, 55112, 55131, 55132, 55121, 55122

TABLE B.10 CROSS REFERENCE --- MESSAGES TO DIAGNOSES TO TESTS

SUMMARY CROSS-REFERENCES, FILE: XREF2 --- AFFECTED-COMPONENTS <=> DIAGNOSES <=> TESTS

AFFECTED-COMPONENT	DIAGNOSES	TEST-MODULES
1: O(R1)	10	55111, 55131
2: S(R1)	2	55111
3: O(R2)	10	55111, 55131
4: S(R2)	11	55132
5: O(R3)	9, 7	55131, 55132
6: S(R3)	10	55111, 55131
7: O(R4)		
8: S(R4)	10	55111, 55131
9: O(R5)	10	55111, 55131
10: S(R5)	11	55132
11: O(R6)		
12: S(R6)		
13: O(R7)		
14: S(R7)		
16: S(R8)		
15: O(R8)	7	55132
17: O(R9)	8	55111, 55112
18: S(R9)	8	55111, 55112
20: S(R10)	8	55111, 55112
19: O(R10)		
21: O(R11)	8	55111, 55112
22: S(R11)	2	55111
23: O(C1)		
24: S(C1)	7	55132
25: O(CR1)	7	55132
26: S(CR1)		
28: S(CR2)		
27: O(CR2)	10	55111, 55131
29: O(CR3)	3	55112
30: S(CR3)	2	55111
32: S(CR4)		
31: O(CR4)	4	55121
33: O(CR5)	5	55122
34: S(CR5)		
03500: ON(Q1A)	10	55111, 55131
03600: OO(Q1A)	10	55111, 55131
03700: OS(Q1A)	10	55111, 55131
03800: NS(Q1A)	10	55111, 55131
03900: SS(Q1A)	7	55132
04000: SN(Q1A)	7	55132
04100: SO(Q1A)	12	55111, 55131, 55132
04200: NO(Q1A)	10	55111, 55131
04300: ON(Q1B)		
04400: OO(Q1B)		
04500: OS(Q1B)		
04600: NS(Q1B)	2	55111
04700: SS(Q1B)	10	55111, 55131
04800: SN(Q1B)	10	55111, 55131
04900: SO(Q1B)		
05000: NO(Q1B)	7	55132
05100: ON(Q2)		
05200: OO(Q2)	7	55132
05300: OS(Q2)		
05400: NS(Q2)		
05500: SS(Q2)	10	55111, 55131
05600: SN(Q2)	10	55111, 55131

TABLE B.11 CROSS-REFERENCE --- AFFECTED COMPONENTS TO DIAGNOSES TO TESTS

05700: SO(Q2)	11	55132
05800: NO(Q2)	7	55132
05900: ON(Q3)		
06000: OO(Q3)	7	55132
06100: OS(Q3)		
06200: NS(Q3)		
06300: SS(Q3)	6	55131
06400: SN(Q3)	6	55131
06500: SO(Q3)	7	55132
06600: NO(Q3)	7	55132
06700: ON(Q4)	8	55111, 55112
06800: OO(Q4)	8	55111, 55112
06900: OS(Q4)	8	55111, 55112
07000: NS(Q4)	8	55111, 55112
07100: SS(Q4)	3	55112
07200: SN(Q4)	8	55111, 55112
07300: SO(Q4)	8	55111, 55112
07400: NO(Q4)	3	55112

TABLE B.11 CROSS-REFERENCE --- AFFECTED COMPONENTS TO DIAGNOSES TO TESTS (continued)

SUMMARY CROSS-REFERENCES, FILE: XREF2 --- UUT-CONNECTING-POINTS <=> TEST-MODULES <=> ATE-CONNECTING-POINTS

UUT-CONNECTING-POINT	TEST-MODULES(S/M)	ATE-CONNECTING-POINTS
J1_G/E29V	55111(S), 55111(M), 55112(M), 55112(S), 55121(S), 55122(S), 55131(S), 55132(S)	
J1_B/TANK_POWER_R	55111(S), 55121(M), 55122(M), 55131(M), 55132(M), 55112(S), 55121(S), 55122(S), 55131(S), 55132(S)	
J1_F/E15V	55111(S), 55112(S), 55121(S), 55122(S), 55131(S), 55132(S)	
J1_H/TANK_BATTERY	55111(S), 55112(S), 55121(S), 55122(S), 55131(S), 55132(S)	
J1_A/Q_SWITCH	55111(S), 55121(M), 55122(M), 55132(M), 55112(S), 55121(S), 55122(S), 55131(S), 55132(S)	
S1	55112(S)	
J1_E/AUX_POWER	55131(S), 55132(S)	
J1_240	55131(M)	
J1_2200/LOADRESISTOR	55132(M)	
J1_C/EXT_BATTERY_		

TABLE B.12 CROSS-REFERENCE --- UUT_CONNECTION_POINT TO TEST MODULES TO ATE_CONNECTION_POINT

SUMMARY CROSS-REFERENCES, FILE: XREF2 --- ATE-FUNCTIONS <=> TEST-MODULES

ATE-FUNCTION,TYPE	TEST-MODULES(S/M)
P_SUPPLY/DC_POWER_SUP, S	55111(S), 55112(S), 55121(S), 55122(S), 55131(S), 55132(S)
LOAD_L/LOAD_LINEAR, S	55111(S), 55112(S), 55121(S), 55122(S), 55131(S), 55132(S)
AMP_M/AMPEREMETER, M	55111(M), 55112(M)
COOL/COOLANT, S	55112(S)
VOLT_M/VOLTMETER, M	55121(M), 55122(M), 55131(M), 55132(M)
LOAD_NL/NONLINEAR, S	55131(S), 55132(S)
ABS/ABSOLUTE, E	55111(M), 55121(M), 55122(M), 55131(M), 55132(M)

TABLE B.13 CROSS-REFERENCE --- ATE FUNCTION TO TESTS